版权注意事项：1、书籍版权归著者和出版社所有；
2、本PDF仅用于个人获取知识，进行私底下知识交流；
3、PDF获得者不得在互联网以任何目的进行传播；
如有需要，请尽量购买正版实体书！支持书籍作者！！



移动开发系列

21天精通微信小程序开发

王延平 编著

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

移动开发系列

21天精通微信小程序开发

王延平 编 著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

这是一本帮助编程爱好者和从业人员从零开始学习微信小程序设计的书。本书紧跟微信小程序的技术发展，是目前在该领域率先进行系统的技术探讨和培训的著作。本书站在学习的角度，目的是让学习者通过21天的时间了解到微信小程序的技术框架结构、开发环境搭建、编程语言细节等。本书全面地讲解了微信小程序的逻辑层与视图层的架构特点、JavaScript语言的入门和进阶知识、WXML语言的语法特点、WXML视觉组件的使用、WXSS样式表语言的语法入门、常用样式描述语句、盒模型和Flex布局等，快速具备微信小程序开发的能力。

这是一本从零学习微信小程序开发的专业书，不需要额外的专业基础就能让你读懂这本书。对于初学者，本书可以作为一本系统的学习教程来使用；对于有经验的前端开发者，本书可以作为微信小程序开发的参考书来阅读。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

21天精通微信小程序开发/王延平编著. —北京：电子工业出版社，2017.2
（移动开发系列）
ISBN 978-7-121-30850-5

Ⅰ. ①2… Ⅱ. ①王… Ⅲ. ①移动终端—应用程序—程序设计 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字（2017）第018578号

策划编辑：张迪（zhangdi@phei.com.cn）
责任编辑：张迪
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编 100036
开　　本：787×1092　1/16　　印张：16　　字数：410千字
版　　次：2017年2月第1版
印　　次：2017年2月第1次印刷
印　　数：3 000册　　定价：49.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询服务方式：（010）88254469，zhangdi@phei.com.cn

推荐序：价值观

从2016年11月开始公测以来，微信小程序的上线时间就一直成为众多圈内人士及用户密切关注的话题，在临近新年到来之际答案得以揭晓——2017年1月9日，微信小程序正式上线。

除了揭晓这个答案，在2017微信公开课PRO版这个非常重要的微信集中展示平台，腾讯公司高级执行副总裁、微信事业群总裁张小龙还重点介绍了小程序的发展历程，并全面阐述了其创作理念和开发者最关心的问题。其中的问题包括：有没有小程序的入口？会不会提供类似于小程序商店的地方供用户下载？小程序能不能推送消息？

对以上问题张小龙给出的答案皆为否定——没有！与之前的订阅号一样，用户需要订阅才能进入，在微信中没有专门的入口，小程序则可以通过二维码扫描进入；正如公众号没有公众号中心，微信里也不会做小程序的分发商店，只是可以查找和搜索；小程序不能推送消息，只会提供一些服务触达的方式。而且，除了这三点之外，微信小程序还没有订阅机制，没有粉丝；不能分享到朋友圈，只可以分享到聊天中；不能做游戏；与公众号是独立的关系。对这些设计，从常理来看有些人很不能理解，甚至觉得是贻笑大方，毕竟这几个拓展用户与产品接触面及扩充增值应用的方法是很多产品常用的，并且大多会有不错的实际收效。那么，为何张小龙率领的团队会有自己的打算呢？

在一年前的公开课上，张小龙分享过微信团队四点核心价值观：一切以用户价值为依归；让创造发挥价值；好的产品应该是用完即走的；让商业化存在于无形之中。

其中前面两点应该是所有团队、所有产品都应视为要旨而去遵循的价值观，相信这样的看法不会有太多的异议。毕竟，用户是产品最终的受众，不能深入挖掘用户需求的产品注定不会走远。再者，融入了智慧的创作过程就应该要产生价值，为用户，为合作伙伴，为社会，也为自己，这也是天经地义的。不过，不同的团队往往因为对这两点的理解不同而会有不同的推进方案和结果，也会出现上下落差极大的状况。而微信团队恰恰在这两点上做得最优秀，这也应该是其到目前为止非常成功的重要缘由。

而如果要深究他们为何能做到最优秀，后面的两点价值观则可以更进一步剖析。我们可以将前两点看作方向，是永远不可能达到极致但必须持续向最高远追求的理想，是形而上层面的道！后面两点则在一定程度上我们可以将之看成术，因为它指引了具体执行过程当中如何“为”的问题，所以它又是完全融合了前面所述“道”层面的术。而这正是其最高妙之处，展示了张小龙及团队对商业、用户，以及产品深邃而精准的理解。

“好的产品应该是用完即走的”，“让商业化存在于无形之中”，这两点表面上看似乎有悖于常道，但却暗合了道家“无为而治”思想的最高境界。老子《道德经》曾说“无为而无不为”，“无为”并不是说不作为，而是指不执着、不妄为，是要顺着事物的自然规律，

顺着人的原初本性而为，这样能充分发挥万事万物天地自然的生命力而达到其自我实现，以至于全盘实现。所以之后还有“取天下常以无事”的说法，天下都能“无为而治”，团队管理与产品规划更不应是难事了。

关于“好的产品应该是用完即走的”，张小龙提到：“一个好的产品不是黏住用户，而是尽量让这个用户离开你的产品”，因为这对于用户来说是效率最高的结果，是自己最轻松自然的状态，从策划团队的指导思想来说这就是以用户价值为归依的体现，就是真正顺应了人这个本体所具有的本性状态，用户使用其产品自然会身心愉悦而无丝毫不快。在 2017 微信公开课 PRO 版上，张小龙再次着重强调了这个价值观。而且说去年提出了“用完即走”的观念后“一些业内人士都会来笑话我们”，但对此他们仍然岿然不动，因为他们坚信微信只是一个工具，其目的是帮助用户以最高效的方法去完成任务，这是工具的目的，工具的使命。用户完成自己的任务后，自然便不必再停留在产品中。而这一理念也一直贯穿在微信的所有产品线上，所以微信倡导去中心化，针对用户在哪里可以启动小程序这一问题，张小龙表示最希望的启动方式是扫二维码。这样，尽管在微信里面没有专门的入口，而每一个小程序关联的企业、卖场、产品包装、户外广告、影视传媒等都能够轻松显示二维码，这反而形成了“无入口而无处不是入口”的情况。而平常，很多团队的工作都是简单直接地围绕如何黏住用户来思考的，都是希望用户尽可能有更长的持续使用时间，都是以自我价值体现为出发点，以“为”的意识去策划，这样就忘却了产品的主体是用户，违背了用户自然的本性上的需求。没有用户会愿意浪费时间去做很多低效率的事情，哪怕他是在休闲时光享受着休闲的应用。所以就会出现“有为”却“为”不了，而表面上“无为”却大有所为。

“一个好的产品的商业化和用户的价值、用户的体验是并不矛盾，好的商业化应该是不骚扰用户，并且是只触达他需要触达的那一部分用户”，张小龙在阐述“让商业化存在于无形之中”的价值观时提到了这句话。“无形”是很多人看不到的用户真正需要的那些点，而这些点往往是用户最在乎的。“骚扰用户”就是被很多人所反感，往往不会因为一个应用好而忍气吞声，频繁骚扰的结果肯定是被直接卸载。所以小程序的推送也是不被允许的，除非用户自己选择愿意接收。

基于这样的理念和实际开发，如今呈现出来的小程序就是无须安装、触手可及、用完即走、无须卸载。对于用户来说就是使用起来最简单、最快捷，同时对用户对手机也最没有负担。用户使用前后脑子里不需要思考任何东西，想用马上就拿来，用后随即就完事。而对于微信团队来说这样的结果就是最不想黏住用户，反而是黏住用户最多、最不想用户花费更多的时间，但确实会让更多的用户在微信上花更多的时间。所以，小程序也注定不会是“小”程序，它必将成为 2017 年移动互联网领域最热门的话题之一，也将带动大批关联的企业、团队、开发者加入其中，用户也能从微信平台获得更多的实用价值。

基于如此的移动互联网技术演进大势，作者非常合适宜地创作了这册图书《21 天精通微信小程序开发》，出发点就是希望更多的开发者能够加入微信小程序这波洪流，并借此机会能进入更广阔、更深入的开发领域，因此其技术内容的创作及文字语句的写作都是专门针对相对比较基础的开发者。最贴心的是，作者还依照心理学家研究的成果“人养成或改变一个习惯只需要 21 天”来编排，将全部内容按日期分章节来规划阅读者的学习步骤，可谓

考虑得非常周到。

当然，阅读本书的开发者最好也能够让自己所学的过程契合上文所述的“无为而无不为”的境界，于学习途中应忘记这个“21 天”，明白它背后真正的要旨是期望大家能保证持续不间断地阅读及实作，克服怠惰或其他影响意志的不良习性依照本书循序渐进，勿三心二意，勿半途而废，借本次学习机会按照养成习惯的路径，由“刻意而不自然”的学习状态，经过“刻意而自然”直到“不经意，自然”的学习习惯。

在此基础上可以更进一层的是，大家借微信团队开发小程序及整个微信产品的价值观，想想自己需要的学习价值观和人生价值观，想想自己的理想和职业规划。一方面考虑清楚自己为何要阅读这本书，为何要学习小程序。是学得一技专长吗？还是作为一个契机和开始。如果能如此，你收获的一定不仅仅是微信小程序的开发技能，而将会收获到更多的平台和机遇，以及更美妙的未来！

范再生

掌脉科技 COO（原《新潮电子》杂志社主编）

2017 年元旦

前　言

个人移动终端的轻应用时代已悄然来临。

微信小程序，一种依托微信轻应用理念的产品，尚未面世就掀起了一轮一轮的讨论热潮：微信的野心在哪里？小程序能做什么？开发者怎样去抓住小程序的风口？一个个问题摆在开发者面前，本书立足微信小程序已公开的技术架构，从技术和运营层面尝试解答开发者的问题，也尝试着能帮助没有开发经验的朋友，从零开始学习编程，快速掌握微信小程序的开发知识，成为一名“小程序员”。

本书有如下几个特点：

1. 覆盖全面

本书全面介绍了微信小程序前端开发的相关支持，从微信的框架讲起，介绍了 JavaScript、WXML、CSS 等语言的基础知识和它们在微信小程序中的应用。

2. 讲解详细

考虑我们的读者大多数是没有编程经验的朋友，我们对页面、组件、函数、面向对象等入门必须掌握的概念进行了深入浅出的讲解，内容丰富，语言平实，力求让每一位读者都能快速了解前端页面和逻辑开发的方法，走进程序员的大家庭。

时间仓促，加之笔者水平有限，书中难免会出现一些错误或者需要改进的地方，恳请读者批评指正。为了能更好地与您交流，您可以加我的微信或者 QQ 一起讨论小程序有关的话题，我的微信号和 QQ 号都是 2264538870。

在本书撰写的过程中，得到了许多专业的、治学严谨的朋友的支持和鼓励，他们帮我整理思路、收集资料、校对书稿，没有他们的帮助，我也不可能在如此短的时间内完成书稿，在此谨向各位朋友的无私付出说一声谢谢：

陈　波　吴万海　巢　洁　成　雯　谭小璐　丁卓群　熊丹洁　徐　翔　明　鉴　张明彪　洪　森　于莎莎　尹金林　程时丛　赵云飞

感谢电子工业出版社的张迪编辑，感谢她在我的写作中提供的支持，也正因为她的认真与高效这本书才能如此迅速地与大家见面。

感谢腾讯公司的微信小程序团队，是他们创造了这样一款优秀的产品。

谨以此书献给我的家人，以及爱我和我爱的朋友们。

目　录

第1部分　微信小程序

第1天　微信小程序概述 …… 2
1.1　什么是微信小程序 …… 2
1.2　小程序在个人移动互联时代的意义 …… 3
1.3　微信大事记 …… 4
1.4　新的消息 …… 6
第2天　微信小程序的技术架构特点 …… 10
2.1　微信小程序在开发上的特点 …… 10
2.2　微信将会支持的小程序分发模式 …… 11
2.3　微信小程序将会给前端设计带来的变革 …… 11
2.4　学习微信小程序开发的路径 …… 12

第2部分　开发前的准备

第3天　小程序的注册申请 …… 14
3.1　申请微信小程序 …… 14
3.2　微信认证 …… 15
3.3　小程序绑定微信开放平台账号 …… 16
第4天　开发环境的搭建 …… 18
4.1　认识小程序管理后台 …… 18
4.2　小程序开发环境的搭建 …… 19
4.3　初识小程序开发工具 …… 20
4.4　编辑模式 …… 21
4.5　调试模式 …… 22
4.6　项目管理 …… 23

第3部分　微信小程序的技术框架

第5天　框架和配置文件 …… 26
5.1　小程序的目录结构 …… 26
5.2　JSON－小程序的配置文件 …… 27
5.2.1　pages－页面配置 …… 28
5.2.2　window－窗口表现配置 …… 29

5.2.3　tabBar－导航栏配置 …… 29
5.2.4　networkTimeout－通信超时事件配置 …… 30
5.2.5　debug－调试模式配置 …… 30
第6天　逻辑层（AppService） …… 32
6.1　小程序的初始化 …… 32
6.2　页面注册 …… 34
6.3　页面的路由 …… 38
6.4　逻辑层的模块化 …… 38

第4部分　WXML语言与视觉组件

第7天　WXML语言和数据绑定 …… 42
7.1　WXML语法规范 …… 42
7.2　视觉组件 …… 43
7.3　组件的数据绑定 …… 44
7.3.1　数据的简单绑定 …… 44
7.3.2　绑定时的运算 …… 46
7.3.3　绑定时的数据组合 …… 47
第8天　逻辑渲染语句 …… 50
8.1　wx:if条件渲染语句 …… 50
8.2　wx:for列表渲染语句 …… 51
第9天　WXML的高级特性 …… 55
9.1　模板 …… 55
9.2　事件 …… 56
9.3　事件对象 …… 58
9.4　引用 …… 61

第5部分　WXSS样式表语言

第10天　WXSS语言基础 …… 64
10.1　WXSS（CSS）简介 …… 64
10.2　CSS基本语法 …… 65
10.3　选择器 …… 66
10.3.1　id选择器 …… 66
10.3.2　class选择器 …… 67
10.4　使用CSS样式表 …… 67
10.4.1　内联样式 …… 68
10.4.2　外部样式表 …… 68
10.5　WXSS的新增特性 …… 69

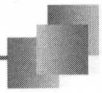
第 11 天　CSS 常用属性 …… 71
11.1　尺寸属性 …… 71
11.2　背景属性 …… 72
11.2.1　背景颜色 …… 72
11.2.2　背景图像 …… 72
11.3　文本格式 …… 72
11.3.1　文本颜色 …… 72
11.3.2　文本对齐方式 …… 73
11.3.3　文本转换 …… 73
11.3.4　文本缩进 …… 73
11.4　字体属性 …… 74
11.4.1　字体系列 …… 74
11.4.2　字体大小 …… 74
第 12 天　CSS 布局属性 …… 75
12.1　盒模型（Box Model） …… 75
12.2　显示方式与可见性 …… 76
12.3　position 定位 …… 77
12.4　float 浮动属性 …… 78
12.5　边框 …… 78
12.5.1　边框样式 …… 78
12.5.2　边框宽度 …… 79
12.5.3　边框颜色 …… 79
12.6　外边距和内边距 …… 80
第 13 天　Flex 布局 …… 81
13.1　Flex 布局简介 …… 81
13.2　模型 …… 81
13.3　容器属性 …… 82
13.3.1　主轴方向属性 …… 82
13.3.2　换行属性 …… 83
13.3.3　简写形式 …… 83
13.3.4　主轴对齐属性 …… 84
13.3.5　侧轴对齐属性 …… 84
13.3.6　多行对齐属性 …… 85
13.4　项目的属性 …… 85
13.4.1　顺序属性 …… 85
13.4.2　放大比例属性 …… 85
13.4.3　缩小比例属性 …… 86
13.4.4　伸缩基准属性 …… 86
13.4.5　简写形式 …… 86
13.4.6　单项对齐属性 …… 87

第6部分　JavaScript 入门

第 14 天　JavaScript 语言基础 …… 90
14.1　文件的编辑和运行 …… 90
14.2　语句 …… 91
14.2.1　语句块 …… 91
14.2.2　注释语句 …… 92
14.2.3　小程序的调试输出语句 …… 92
第 15 天　变量与数据类型 …… 93
15.1　变量 …… 93
15.2　操作符 …… 94
15.3　数据类型 …… 95
15.3.1　数值 …… 95
15.3.2　字符串 …… 95
15.3.3　布尔型 …… 96
15.3.4　数据类型转换 …… 96
15.3.5　null 和 undefined …… 97
15.4　数组 …… 97
15.4.1　创建和初始化数组 …… 97
15.4.2　数组元素的存取 …… 98
15.4.3　数组的操作 …… 98
第 16 天　流程控制语句 …… 100
16.1　条件语句 …… 100
16.1.1　if 语句 …… 100
16.1.2　条件运算符 …… 101
16.1.3　逻辑操作符 …… 102
16.1.4　switch 语句 …… 102
16.2　循环语句 …… 103
16.2.1　for 循环 …… 103
16.2.2　for in 循环 …… 104
16.2.3　while 循环 …… 104
16.2.4　do while 循环 …… 105
16.2.5　break 退出循环语句 …… 105
16.2.6　continue 继续循环语句 …… 106

第7部分　JavaScript 进阶

第 17 天　面向对象程序设计（OOP） …… 108
17.1　类和对象 …… 108



17.2　创建对象 …… 110
17.2.1　直接创建 Object 类的对象实例 …… 110
17.2.2　用 new 关键字创建对象 …… 111
17.2.3　this 关键字 …… 111
17.3　使用对象 …… 111
17.4　继承 …… 112
第 18 天　函数 …… 114
18.1　定义函数 …… 114
18.1.1　定义命名函数 …… 114
18.1.2　定义匿名函数 …… 115
18.1.3　用 Function 类创建匿名函数 …… 116
18.2　函数的参数 …… 116
18.3　函数的返回值 …… 117
第 19 天　JSON 语法详解 …… 118
19.1　JSON 的语法结构 …… 118
19.2　使用 JSON …… 119
19.2.1　从 JSON 中读数据 …… 119
19.2.2　向 JSON 中写数据 …… 120
19.2.3　修改 JSON 中的数据 …… 120
19.2.4　删除 JSON 中的数据 …… 120
19.2.5　遍历 JSON 对象 …… 120
第 20 天　ES6 语法 …… 122
20.1　ES6 简介 …… 122
20.2　ES6 常用语法详解 …… 122
20.2.1　let 变量声明语句 …… 122
20.2.2　const 常量声明语句 …… 123
20.2.3　类操作语句 …… 123
20.2.4　arrow function 箭头函数 …… 124

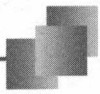
第 8 部分　工具和社区

第 21 天　开发工具和学习社区 …… 126
21.1　Coolsite360 可视化工具 …… 126
21.2　即速应用 …… 129
21.3　开发社区介绍 …… 130

附录　微信小程序工具书

附录 A　小程序组件详解 …… 134
A.1　视图容器 …… 134

A. 1. 1　view …… 134
A. 1. 2　scroll – view …… 135
A. 1. 3　swiper …… 137
A. 2　基础内容组件 …… 138
A. 2. 1　icon …… 138
A. 2. 2　text …… 140
A. 2. 3　progress …… 140
A. 3　表单组件 …… 141
A. 3. 1　button …… 141
A. 3. 2　checkbox – group …… 143
A. 3. 3　form …… 144
A. 3. 4　input …… 145
A. 3. 5　label …… 148
A. 3. 6　picker …… 152
A. 3. 7　radio – group …… 154
A. 3. 8　slider …… 155
A. 3. 9　switch …… 156
A. 3. 10　textarea …… 157
A. 4　导航组件 …… 158
A. 5　媒体组件 …… 160
A. 5. 1　image …… 160
A. 5. 2　video …… 162
A. 6　地图组件 …… 165
A. 7　画布组件 …… 166
附录 B　小程序 API 详解 …… 169
B. 1　网络 …… 171
B. 1. 1　发起请求 …… 172
B. 1. 2　上传、下载 …… 173
B. 1. 3　Web Socket …… 175
B. 2　媒体 …… 178
B. 2. 1　图片 …… 178
B. 2. 2　录音 …… 180
B. 2. 3　音频播放控制 …… 181
B. 2. 4　音乐播放控制 …… 183
B. 2. 5　音频组件控制 …… 185
B. 2. 6　视频 …… 187
B. 2. 7　视频组件控制 …… 188
B. 3　文件 …… 190
B. 4　数据缓存 …… 193



B.5　位置 …… 198
B.5.1　获取位置 …… 198
B.5.2　查看位置 …… 199
B.5.3　地图组件控制 …… 200
B.6　设备 …… 202
B.6.1　系统信息 …… 202
B.6.2　网络状态 …… 203
B.6.3　重力感应 …… 203
B.6.4　罗盘 …… 204
B.6.5　拨打电话 …… 204
B.6.6　扫码 …… 205
B.7　界面 …… 205
B.7.1　交互反馈 …… 205
B.7.2　设置导航条 …… 208
B.7.3　导航 …… 209
B.7.4　动画 …… 210
B.7.5　绘图 …… 214
B.8　开放接口 …… 214
B.8.1　登录 …… 214
B.8.2　用户信息 …… 220
B.8.3　微信支付 …… 221
B.8.4　模板信息 …… 222
B.8.5　客服消息 …… 226
B.8.6　分享 …… 234
附录 C　JavaScript 的关键字和操作符 …… 235
C.1　JavaScript 操作符 …… 235
C.2　字符串方法（表 C-8） …… 237
C.3　Math 对象 …… 237
C.4　Date 对象（表 C-13） …… 238

第1部分
微信小程序

微信已经成为我们生活中密切相关的移动应用，据统计，微信活跃用户已经超过6亿，覆盖到了200个以上的国家，很多人的关系链已经从手机通讯录转移到了微信，人们用微信发语音聊天、刷朋友圈，走路在用微信，乘车在用微信，甚至吃饭、娱乐都会拍照上传到微信朋友圈。

对我们开发者有利的是，微信这个移动互联网上的重要入口，一直都是以开放的姿态面世的。微信提供了公众平台和开放平台用于功能性应用的开发，现在又进一步提供了微信小程序的开发接口，为开发者提供了广阔的想象空间和难得的商业机遇。

本章我们将用两节课的时间对微信小程序做一个介绍，让大家对微信小程序的历史和它将要带来的商业机遇，以及微信小程序在开发上的特点，微信小程序能做什么有一个了解，避免大家在产品设计和开发上走弯路。



本章要点

- 微信小程序的商业价值
- 微信小程序的技术架构特点
- 微信小程序的学习路径

第 1 天

微信小程序概述

1.1 什么是微信小程序

关于微信小程序，张小龙在朋友圈里是这样描述的：

> 微信小程序是一种不需要下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户扫一扫或搜一下即可打开应用。也体现了“用完即走”的理念，用户不用担心是否安装太多应用的问题。应用将无处不在，随时可用，但又无须安装卸载。

微信小程序与微信公众平台的订阅号、服务号、企业号一样，属于并行的微信公众号体系。

微信小程序不须要下载和安装即可在微信里启动和使用。

结合目前微信官方公布的技术接口来看，微信小程序是微信提供的一个定制 Web 页面系统，微信在小程序里提高了对 H5 特性的支持能力，开放了更多的系统调用。例如，GPS、传感器等接口的调用等。

由于微信小程序针对微信浏览器做了专门的优化，和 H5 页面比起来将有更好的用户体验、更快的加载速度和更多的功能。微信小程序的服务架构如图 1-1 所示。

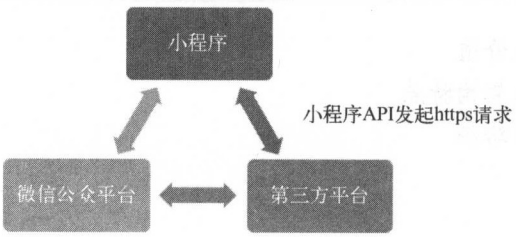


图 1-1　微信小程序的服务架构

个人移动计算时代刚刚来临时，曾经有过一个关于 WebApp 和 NativeApp 孰优孰劣的大讨论，后来以 Facebook 彻底放弃 WebApp 成为讨论的一个终结。NativeApp，也就是我们经常说的“原生应用”赢得了阶段性的胜利，而 H5 技术在发展的同时，受限于浏览器支持和执行效率，有被渐渐边缘化的趋势。



有很多因素决定了近几年 H5 渐渐失势，失去了绝大部分市场份额。其中有一点是 H5 并不能真正做到一次开发多处使用，而是要面临不同浏览器和 App 平台更多兼容性挑战。例如，过去微信的内置浏览器不能支持很多的 H5 特性，这也是微信小程序出现的技术方面的原因。

几年前，当微信订阅号第一次出现在用户面前时，我们认为微信带给了 H5 一个独特的发展机会，所有 H5 产品将以微信的接口为标准开发，浏览器逐步向微信标准靠拢，最终实现一次开发多处使用的目标。

今天来看，为这一理念而生的微信小程序终于还是来了。距离 H5 产品的标准化运行环境又近了一步，在微信倡导的规范化框架里，微信里 H5 产品质量也会大大提高，这是可以预见的。

1.2　小程序在个人移动互联时代的意义

相比微信公众号，小程序提供了更丰富的框架组件和 API 接口供开发者调用，其中包括界面、视图、内容、按钮、导航、多媒体、位置、数据、网络、重力感应等。开发者可以通过这些组件和接口上线非常流畅的应用。

微信小程序会挑战 App 吗，这个话题目前是大家最关心的。但是问题的答案可能会令大家失望。

随着移动产品的发展，H5 和 App 的产品定位已经非常清晰，用户的使用习惯已经形成，这个格局将不会因为小程序的出现而发生巨大的改变。

比较 App 的形式，微信小程序有开发门槛低、开发效率高、捆绑微信平台易于传播的优势，但同时也有与 H5 类似的劣势，就是执行效率低，无法实现对人机交互和多媒体要求比较高的应用，如网络游戏等。

H5 和 App 的区分界限已经不再单纯是用户体验问题，大量的 App 在采用 Native + H5 开发模式。App 和 H5 产品的主要区别是对于产品分发的需求。

在现有的格局下，App 产品通过 Apple AppStore，以及谷歌市场和手机应用商店等分发渠道的榜单、搜索及社交产品与广告平台等方式被用户发现、下载、安装。而 H5 产品主要靠手机网站和社交平台的分享被用户发现和使用。

目前 App 获取用户的门槛很高，高门槛带来的好处是，用户的二次消费门槛变低了。H5 在分发上的优点是用户发现和安装使用的门槛低，然而后续消费门槛却非常高。而这方面正好是 App 的优势。

事实上到目前为止，App 的用户规模和活跃度总体还是远远高于公众号的。不要因为微信活跃度高就认为公众号或者小程序的活跃度就会高了。如果这样类比，小程序和微信活跃度的关系，应该对应 App 和手机活跃度的关系上了。显然这不是合理的数据分析方式。

小程序出现以后，很多人觉得微信的未来目标就是成为手机上的操作系统。这也许是腾讯和微信的最终定位，小程序也彰显了腾讯的野心，但是目前讨论这个问题还为时太早。

我们知道，手机操作系统的核心是应用的管理，而微信是聊天窗口。这个本质不改变，就很难改变“应用 App”和“小程序”在各自体系里的地位。

短期内，用户的操作习惯不会被改变，手机操作系统将会依然被 App 主导，侧重重度体验、交互性、高粘性需求的产品还是会以 App 的形式分发，这是由 App 的高执行效率决定的。

微信平台上订阅号、服务号、企业号和小程序将依赖推送、扫描和社交传播渠道来搭建。微信小程序将会有微信推荐、搜索、二维码扫描和社交传播几种方式，推送将保留给微信的传统公众号，而小程序可以嵌入到公众号中传播。

过去成功的公众号基本上也就是靠内容运营或者线下场景扫描来实现快速传播，如果具备这两方面的资源，又没有系统资源的使用必要，无论是不是有小程序出现，都早就不须做 App 了。

1.3　微信大事记

1. 2010 年

2010 年 12 月，小米科技发布手机端即时通信工具“米聊”，在国内语音 IM 市场上抢得了先机。米聊主打熟人圈，这一定位限制了米聊用户的发展。360 创始人周鸿祎预言，米聊将会为腾讯做嫁衣。

2. 2011 年

2011 年 1 月 21 日，微信 1.0 发布。该版本支持导入 QQ 现有的联系人资料，但仅有即时通信、分享照片和更换头像等简单功能，因此并不为外界所看好。

2011 年 1 月至 4 月，微信 1.1、1.2、1.3 版本相继发布，增加了对手机通讯录的读取权限，与腾讯微博私信的互通，以及多人会话功能的支持，获得了四五百万的注册用户。

2011 年 5 月 10 日，微信 2.0 发布，新增了语音对讲功能，这一举措使得微信的用户群第一次有了显著增加并且击败了国内所有同类产品。

2011 年 8 月，微信 2.3 版本中增加了“查看附近的人”功能，微信开始主打陌生人社交，再一次引爆了微信用户的增长点，此时微信用户已达 1500 万。

2011 年 10 月 1 日，微信 3.0 版本加入了“摇一摇”和“漂流瓶”功能，增强了陌生人社交能力，同时增加了对繁体中文语言界面的支持，增加了香港、澳门、台湾地区和美国、日本两个国家的用户手机号绑定服务，微信开始迈出了国际化的步伐。

2011 年 12 月 20 日，微信 3.5 发布，推出二维码名片，方便用户通过扫描或在其他平台上发布二维码名片，拓展微信好友。

3. 2012 年

2012 年 4 月 19 日，微信发布 4.0 版本，推出了微信朋友圈，进一步加强了微信的用户黏度。同时，微信 4.0 的官方英文名称被定为 WeChat，进一步加快了微信的国际化进程。

2012 年 7 月 19 日，微信 4.2 版本增加了视频聊天插件，发布了网页版微信界面。从此微信不再仅仅是一款手机即时通讯客户端，它开始把触角伸向了桌面领域。

2012 年 8 月 23 日，微信公众平台上线，公众平台的内容能被用户分享到微信朋友圈，用于增强用户黏性。

2012 年 9 月 5 日，微信 4.3 版本增加了摇一摇传图功能，该功能可以方便地把图片从电脑传送到手机上，此版本还新增了语音搜索功能，进一步方便了用户对微信信息的把控。

4. 2013 年

2013 年 2 月 5 日，微信 4.5 发布，该版本支持实时对讲和多人实时语音聊天，支持对聊天记录进行搜索、保存和迁移。同时，还加入了语音提醒和根据对方发来的位置进行导航的功能，从中可以看到微信将往移动秘书方向发展。

2013 年 3 月 19 日，公众平台开放自定义菜单的内测资格申请。

2013 年 4 月 11 日，腾讯微信团队发布声明，称网上流传的微信要对用户收费消息系恶意造谣，同时强调微信绝不会对用户收费。

2013 年 8 月 5 日，微信 5.0 发布，公众平台将公众号细分为服务号和订阅号。服务号每月只能群发 1 条消息，订阅号每天可发 1 条。微信还在这个版本中添加了表情商店和游戏中心，支持微信支付，微信开始增强商业化能力，凸显出挑战支付宝的野心。

2013 年 8 月 19 日，网易和中国电信联合发布通信社交工具“易信”，此应用实现了跨网发送免费短信和免费电话留言功能，外界普遍认为用户需求不强烈，很难对微信构成威胁。

2013 年 8 月 29 日，微信公众平台发布数据统计功能。

2013 年 9 月 23 日，阿里巴巴发布移动好友互动平台“来往”，其核心功能是实现熟人之间的社交，主打私密安全牌，借微信陌生人交友存安全隐患的弱点切入市场。

2013 年 10 月 29 日，公众平台新版公测，开放了高级接口。

2013 年 12 月，公众平台发布了一系列公告，打击违法违规的内容和行为，加强了微信公众平台的内容监管。

5. 2014 年

2014 年 1 月 26 日，微信 5.2 发布，该版本增加了微信支付场景，新增“滴滴打车”和“微信红包”功能，微信跟支付宝比起来，杀手锏是基于关系链的支付场景，而非支付功能本身。

2014 年 3 月 5 日，公众平台发布微信支付申请指引，标志着微信支付正式对服务号开放。

2014 年 4 月 4 日，公众平台发布《微信公众平台运营规范》，明文标明微信公众平台反对和禁止的行为及处罚机制与举报机制。

2014 年 4 月 15 日，公众平台进行服务号群发策略调整，所有服务号的群发次数由原来的每月 1 次改为每月 4 次。此外对已微信认证的服务号，开放公众平台高级群发接口，开发者可以通过高级群发接口实现更灵活的群发。

2014 年 5 月 6 日，腾讯公司宣布成立微信事业群（Weixin Group，简称 WXG），腾讯公司高级执行副总裁张小龙出任微信事业群总裁，这意味着微信的发展进入一个崭新的阶段。

2014 年 5 月 9 日，公众平台新增投票和多客服功能。

2014 年 5 月 22 日，微信认证拆分为资质审核和名称审核两个部分。

2014 年 5 月 29 日，公众平台增加微信小店功能，已接入微信支付的服务号可以申请开通微信小店功能。

2014 年 6 月 9 日，搜狗搜索宣布正式接入微信公众号数据，一定程度上帮助微信从封闭走向开放，让微信公众号中产生的内容可以得到更广泛的传播。

2014 年 6 月 6 日，公众平台发布清理集赞行为的公告，对利用朋友圈“集赞”的行为进行打击，处罚力度很大，公众号累计发现四次有集赞行为，永久封号，不可解封。

2014 年 7 月 23 日，微信公众号新增图文阅读次数展示，图文消息底部新增点赞按钮。

2014 年 8 月 14 日，微信 5.4 版本增加了对公众账号及公众账号文章的搜索，意味着用户能从微信信息大海中找到自己想要的信息，构建完整的搜索生态。

2014 年 8 月 27 日，微信公众平台向认证服务号开放模板消息。模板消息仅用于公众号向用户发送重要的服务通知，只能用于符合其要求的服务场景中，如信用卡刷卡通知、商品购买成功通知等，不支持广告等营销类消息，以及其他所有可能对用户造成骚扰的消息。

2014 年 10 月 1 日，微信 6.0 上线，该版本实现了从图文信息到视频信息的进化，并推出了卡包功能，继续加强其商业属性。

2014 年 12 月 12 日，陌陌科技登陆纳斯达克，陌陌成为我国除微信外获得最大成功的社交软件。

2014 年 12 月 19 日，微信公众号开始内测评论功能。此举引起各方舆论的广泛关注和热议，对于放开评论功能利弊与微信未来发展走向的讨论络绎不绝。

6. 2015 年

2015 年 1 月 9 日，微信开放微信内网页开发工具包（微信 JS－SDK）。JS－SDK 增强了 H5 的功能，为微信用户提供更优质的网页体验。

2015 年 2 月 12 日，公众平台全面开放自定义菜单。

7. 2016 年

2016 年 9 月 22 日，微信宣布“应用号”开始内测，定名为“微信小程序”。

2016 年 11 月 3 日，微信团队宣布微信小程序正式开放公测。

2016 年 11 月 19 日，张小龙在朋友圈发布小程序最新进展，从公布的图片来看，微信小程序具备桌面图标方式的入口。

1.4　新的消息

2016 年 12 月 28 日上午，正当本书完稿之际，腾讯的官方公众号“微信公开课”发布了关于小程序新的消息。

文稿以“张小龙首次全面阐述小程序，定档 1 月 9 日上线（内附演讲全文）”为题，明确地公布了微信小程序的计划上线日期。

文稿回顾了小程序一年的发展历程，阐述了小程序的产品理念，更重要的是回答了更多



大家关心的关于小程序的问题。下面是张小龙演讲中关于大家关心的问题的解答原文，我们附上，供大家学习小程序时参考。

● 第一个问题是，小程序的入口在哪里？

小程序在微信里是没有入口的，很多人看到微信开始内测小程序，大家说这是一个新的机会，我们应该第一波上去，去获得一些流量上的红利，但是那就非常遗憾，公众号在微信里面其实也没有入口，一个用户如果没有订阅过任何一个公众号，他在微信里面找不到这样一个入口，小程序也是一样的。如果一个人没有去运行过任何一个小程序，他在微信里也是找不到小程序的入口的。这跟之前提到的一些产品理念相关，在微信里我们一直在倡导去中心化的结构，所以你到现在都不可能看到在微信里会有一个订阅号的入口，里面有一个分类，有排序或者有推荐这样的东西存在，这一点从公众号的第一步就坚持是这样的，这个对公众号平台带来了很大的好处，因为当微信没有这样一个入口的时候，所有提供公众号服务的企业他们会想办法把自己的二维码铺到所能够铺的地方去，那就真正实现了公众号的入口其实不是在微信里面，而是在二维码里。如果是我们有一个基础的入口的话，那大家可能抢夺的都是微信里的入口，如果我们有推荐的话也不是一个好事情。我们的推荐就会变得有更多人订阅。

所以对于小程序来说，可以想象一下未来的小程序用户更多是从哪里去启动它？我们更多的是希望小程序的启动来自于扫二维码。前不久跟一个合作伙伴的公司在聊这个话题，他们希望能够知道小程序的发布时间，好做一些准备。他们提出一个场景我觉得特别的切合，他们说现在汽车票其实没有电子化，所有人去坐汽车的话必须要去汽车站现场去买一张票，这是一个很痛苦的过程，你要去排队买票，然后再去坐车，他们希望用小程序来解决这个问题，只需要在每一个汽车站立一个二维码，所有到汽车站的人扫一下二维码就启动购票的小程序，然后直接通过小程序来买好票，这样售票窗口就不用存在了，我认为这是一个非常贴合小程序的想法。

这样的想法会有很多很多，昨天跟我们小程序这一块的同事在聊，他们说有人做了一个小程序特别方便，是在公交站里面等公交站的时候，想要知道下一班车什么时候来，这个时候只要扫一下公交站的二维码，启动公交站的小程序就可以看到下一班车什么时候来，这也是特别典型的一个小程序的场景。

所以至少在前期我们会更多的鼓励小程序以二维码的形式出现在每一个地方，就像公众号的早期一样。

● 第二个问题是，会不会有一个类似于小程序商店的地方，可以去下载小程序？

其实是不会有的，大家可以推理出来，我们没有下载过程，所以不存在一个应用商店去下载。但是大家会说我想在里面去浏览，去找一些小程序，查找是另外一回事，搜索是另外一回事，但是我们并不会存在一个小程序的商店，所以我们也并不会像外界所猜测的那样，做一个 App 的分发想法。从一开始我们就没有这个想法，就像公众号从来不会有一个公众号中心，里面可以分门别类列出所有的东西出来，不会有这样的东西，我们认为应该没有一个中心入口，是去中心化的形态，所以我们也不会做小程序的分类、排行、推荐。

关于推荐大家可能会有一些自己的想法，大家可能会认为为什么不把用户感兴趣的一些东西推荐给他？其实我们也可以把这个想法用到公众号里，我们不会做这样的事情，将来也不会做这样的事情。

在微信里我们曾经考虑过对机器推荐和一个人的社交推荐的差异，朋友圈其实是类似于社交推荐，你每天会在朋友圈看到很多的文章，你的朋友其实起到了一个推荐器的作用，使得你不用依赖系统的推荐。如果我们系统来做，可能没有你的朋友们做得好，因为系统不会给你推荐一些你没有接触过的东西，系统只会强化你接触过的信息，并且不断地去学习你的历史，往你的历史方面继续推荐，但是你的朋友可能在朋友圈里面说，某一部电影很好看，那你会因为你的朋友去看了这部电影而去看这部电影，机器是无法理解这一点的，机器只会采集你过去看过的电影，然后把你看过电影的类型整理出来，认为你就喜欢这一类型的电影，但是你朋友推荐的电影可能是你完全没有接触过的电影形态，所以微信里更多是依赖于社交推荐。小程序这里也是一样的，你可能会使用千奇百怪的小程序，但是我们不会因为你已经使用了一款，如学英语的小程序，就反复给你推荐学英语的小程序，这个话题也不在这里展开了。

- 第三个问题是，小程序与订阅号的关系？

相信很多人会关注这样一个问题，如果我们做一个公众号，我们会收获到很多的订阅用户（即粉丝），并且大家会从粉丝数来判断自己这个号的价值，但是小程序不会像订阅号，所以如果你做一个小程序，不能依赖于我的目的是要收获多少粉丝，粉丝并不意味着访问量，并不是说你有足够多的粉丝你就有足够大的访问量。大家可以回想一下刚才的例子，在 PC 互联网时代，你访问一个网站并不需要你成为这个网站的粉丝，并不需要你去订阅它，如果所有的网站需要你订阅了才能看它的信息，网站可能早就已经死掉了，所以对于小程序来说也是一样的，它只有一种访问的关系，而不是一种粉丝的关系，所以它跟公众号从本质上来说是截然不同的，小程序不是一种公众号，它是一种新的形态。

- 第四个问题是，小程序能不能推送消息？

可能也是很多人非常关注的，答案是不能，因为如果小程序能推送消息，那就意味着你访问的每一个网站都会发一条消息给你，那你可能会崩溃掉，所以它是不能推送消息的，但是我们会提供比较有限的服务触达能力，你以前访问一个网站的时候可能会留下自己邮件地址，当你要的服务有的话这个网站会给你发一个邮件，小程序也类似，当你在小程序里面做了一个操作，并且希望收到后续通知的时候，我们会提供这样一个通知的机制，使你可以通过小程序给你的用户发送一条他所需要的通知，但这个通知不是说谁来过的小程序就能获得一条推送，而是说这个用户在小程序里面主动地确认了他需要获取一条后续的通知，所以它是一种很有限的通知的能力。看起来我们对这种通知的能力限制的非常严格，其实这也是没有办法的事情，就像我刚才举的一个例子，你在一个网站里面留一个地址，你就会收到很多垃圾邮件，我们并不希望这样的事情在小程序里面再次发生。

- 第五个问题是，小程序能不能够分享？

当然有一个问题就不列出来了，小程序能不能分享到朋友圈？肯定不能，这就不作为一个问题了。但是小程序可以分享到聊天里面，可以分享到群里面。其实分享到群里面和聊天里面是有非常大的一个想象空间的。我记得我们内部在讨论这个功能的时候，把一个小程序分享到一个群里面的时候，其实刚开始的形态是非常简陋的，就是我把一个小程序发到一个群里面而已，那只是利用群做了一个传播，让这个群里面的使用这个小程序。但是其实小程序的分享不仅仅是这样子，小程序的分享更多的是希望能够带来一种新的协作方式。

举一个例子，当我把一个投票的小程序发到群里的时候，意味着群里面的每个人可以立

即启动这个小程序，并且利用投票，每个人可以看到其他人的投票。对于一个群来说，这个小程序带有每个人的登录状态，大家访问的是同一个小程序的任务。基于这样群的任务，它可以被群里面的所有人共享，当任何一个人更新群里小程序状态的时候，群里其他人都是可以看到的，基于这个想法你可以想象得到，可能会存在非常多的一些协作式的小程序。

我们还提出了关于小程序页的概念，比如说一个股票的小程序，我分享的只是我当前所看到的 0700 股票的这一页，然后分享到一个群里面去了，群里面的人看到的也是 0700 这样一个股票的页面，我们把它叫作小程序页。这里我们希望的是，我分享到群里的是一个活的数据，是当前我看的信息。我的意图并不是说把一个程序分享到群里面让大家重新运行这个程序，我只是分享了一个活的信息过去，而且在未来我们更希望的是，当然现在还没有做到，我们更希望的是我分享到群里面这一页的信息它是活的，所谓活的意思就是当它出现在一个聊天里的时候，你甚至不用点进去你就能看到这个小程序的表现。例如，我分享一个时钟的小程序到群里面，那么群里面每个人看到这个小程序，不用点进去就可以看到已经有一个时钟在那里运转。暂时这个形态还没有实现出来，但是我个人非常期待。我相信这种协作式的任务，对于小程序的分享会起到一个很大的帮助，我们可以在里面构思出非常多的需要群组一起完成任务的小程序。

- 第六个问题是，很多人会问小程序能不能做游戏？

我们现在并不能。

- 第七个问题是，小程序能不能被搜索到？

会有非常多人说我们早一点做小程序，然后就可以很好地利用这样一个前期的流量，这里特别遗憾地说，我们更希望小程序不是基于一种流量分发的方式获取用户，就像刚才说的我们更希望小程序是通过一种用户触达的方式，当用户需要的时候触达到它，然后使用它，而不是不需要的时候推荐给他使用，所以用户能搜到小程序，但是我们可以极力限制它搜索的能力，避免它滥用，使得用户在微信里面能够搜索得到他需要的一个小程序。

- 第八个问题是，小程序和公众号的关系是怎么样的？

刚才大家明白了小程序和公众号是一种很独立的关系，但是因为很多的公众号和小程序他们可能是一个企业开发的，所以它们应该有某种关联，目前我们提供的一种关联是，你在公众号里面可以看到这个公众号同一个企业还做了哪些小程序，或者你在一个小程序里面你也可以看到，做这个小程序的企业还做了哪些公众号，他们是可以互相跳的。

还有一个小问题是，既然小程序会突出二维码，那么微信里面会不会对于线下的店会有一些提示，在目前这一阶段我们可能会很轻量地让用户能够看到在他附近还有哪些小程序存在，我们说的附近的小程序存在是指附近有哪些在提供服务的店，他们同时也有小程序。比如说在三公里以外有一个士多店，那么你可以看到并立即打开它的小程序，然后买一点什么东西，这是很有可能的。



课后作业

想一想，你认为微信打败 Facebook 和米聊，成为国内用户量最大的即时通信工具的原因，把他写下来。

第2天

微信小程序的技术架构特点

2.1 微信小程序在开发上的特点

腾讯为微信小程序的工程项目开发了完整的编辑开发、运行调试、打包发布环境，形成了一个完整的 IDE。

有些朋友认为微信小程序就是 H5，其实并不是这样。

小程序借鉴了很多前端开发的技术理念，它用 React 实现了“视觉组件”，它用 CMD 的 require 作为面向对象的 JavaScript，用 Vue 实现了标签式逻辑与数据绑定。

小程序用 JavaScript 语言、XML、CSS 语言编写程序代码，写小程序代码几乎与 Web 前端开发完全一样，一个有经验的 Web 前端程序员只需要花费半小时就能快速上手小程序开发，但小程序并不是标准化的 H5 + CSS3 + JavaScript 架构，它和 Web 架构基于的 W3C 规范没有任何关系，小程序使用腾讯全新定义的技术规范和架构，是微信自有的。

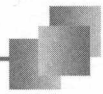
那么小程序与 H5 又有哪些不同呢?

小程序不支持 W3C 的 DOM（文档对象模型）规范，小程序的页面也不是基于 Window、Document 等 JavaScript 内置对象的，所以现有 Web 前端领域的第三方框架，如 Jquery，Zepto、UI 类框架等都不能在小程序里使用。而小程序的 JavaScript 在上下文中自带了 wx 对象，也就是之前公众号开发中 js - sdk 的主对象。

小程序的标签，小程序称为“视觉组件”，并不是基于 HTML 规范的，它是腾讯全新定义的一套完整的标签库，它只能运行在微信的浏览器下，所以我们以往运行在微信服务号、企业号或者通过浏览器访问的前端项目代码，无法直接移植到小程序。

微信小程序定义了自己专有的模型，吸收了主流前端开发中数据、样式、控制逻辑分离的理念，剔除了烦琐的关联配置，并且从规范上要求每个“页面”需要有同名的四个文件：index. js、index. json、index. wxml、index. wxss，各司其职。其中 js 文件采用标准的 JavaScript 语法规范，用于逻辑操作；json 文件基于 JSON 语法规范，用于页面文件的配置；wxml 文件基于 XML 语法规范，用于页面视觉组件的描述；而 wxss 继承了标准的 CSS 语法，用于 WXML 组件样式的定义。

小程序的页面加载基于本地，不需要通过频繁的服务端请求来实现，所有的页面跳转都不需要通过服务端交互。这将使小程序的执行效率大大提高，比服务号、企业号等基于 H5



的 Web 应用模式有更好的用户体验，操作流畅度与反应速度也会更好。这也意味着在没有网络连接的环境下也可以使用微信小程序。结合本地存储，小程序可以满足暂时断网或网络情况较差的场景需求，这是微信服务号和 Web 服务都无法实现的。

微信对于一些通用的用户操作和显示，如等待图标、错误提示、结果展示等，都提出了设计方式上的要求，形成了统一的规范，强制开发者按照规范标准做开发设计，这是微信作为一个准封闭系统，为了用户体验而进行的努力。当然，其意义要从两方面来看，一方面是规范的标准给开发者带来了方便，也让用户使用小程序更高效，但同时也限制了开发者创造性的发挥。

2.2　微信将会支持的小程序分发模式

对于互联网应用产品的开发和运营者来说，如何让用户可以快速找到你的产品，是你最需要关心的事。所有的传播都是为了被用户发现。那么，微信将会给小程序提供怎样的分发模式支持呢?

微信作为一个 IM 平台，目前缺少展示位，很难有足够的位置让用户把常用的小程序展示出来。价值连城的九宫格展示位也已经藏到了系统的三级菜单甚至以后，越藏越深。小程序要获得更好的位置资源确实不易。微信对比浏览器，缺少了域名指向，这让小程序的分发模式又丧失了一种可能性。

现在来看，小程序的入口平台可能会出现在微信首页的导航按钮栏、发现页面或聊天页面。除此之外，大量小程序的分发将会集中在微信的搜索结果里，如何设置小程序的搜索标签属性，让小程序得到更多的展示机会，这会涉及很多的规范性要求，这将会是微信小程序运营者重点思考的问题。

另外，从目前微信流出的一些小程序“谍照”来看，微信小程序将会具备手机桌面图标方式的启动入口，在微信的“发现”页面，也会有小程序的一个统一入口，和“朋友圈”、“附近的人”等并列。

2.3　微信小程序将会给前端设计带来的变革

微信小程序的价值，将会体现在下面几个方面：

(1) 对于开发者，小程序因为兼容 JavaScript 和 XML、CSS 语法规范，这将会使开发门槛更低，开发一个程序将会变得更简单。在小程序平台上线初期，会产生一个密集的应用分发高潮，并将持续半年到一年的时间，开发者将能够借助微信平台获得较大的流量，这将比 App 的流量获取更加容易，也能让营销成本变得更低。

(2) 对用户来说，小程序因为它的即开即用特性，将会减轻手机的应用压力，避免资源浪费。微信小程序的审核机制会比 App 更为严格，应用程序将很难获取到用户的敏感权限，这将使用户使用手机更为安全，小程序的获取渠道将更多地集中在微信上，降低了用户获取应用的时间成本。

基于此，我们可以设想一下，微信小程序正式上线后，将会发生什么：

（1）需要下载 App 才能使用的高频工具类应用，如天气预报、火车订票、安全期日历、日程表、课程表等，很快就会有小程序的版本面世，如果那些拥有知名产品的公司不去开发，别人就会趁虚而入。

（2）由于目前小程序的重复调用机制还不明确，没有关注机制，以往脸萌、无密等火一把就销声匿迹的应用也会出现在小程序中，但更多会以小额收费的形式，赚一波就走。

（3）对于创新型的垂直社交产品会有一个机会，机会大小得看微信对关系链开放的力度。

（4）微信肯定会投入精力到电商类小程序，如微店。

（5）最终商业价值最大的将会是具备稳定高频次应用场景和用户习惯搜索获取的高流量小程序。

2.4　学习微信小程序开发的路径

可以预见的未来，微信小程序会形成一个应用分发新的渠道，同时微信将会短时间内出现一个流量膨胀的入口，也会涌现出一批“微信小程序员”。

作为开发者，我们要学习一个语言的开发，一个框架的使用，并不是仅仅读一本书那么简单，正确的学习方法是：

- 建立全局观，设定一个学习路径；
- 制订切实可行的学习计划，严格执行；
- 从简单的小程序做起，锻炼开发实战能力；
- 加入与小程序开发有关的 QQ 群，微信群，与有经验的人频繁请教，交流、分享；
- 编写一些有可能有刚需的小程序，经常练手，语言规范、组件的使用、API 的调用需要不断地重复使用才能做到得心应手。



课后作业

1. 参照本书目录，合理安排你的时间，制订一个微信小程序开发的学习计划。
2. 微信小程序用到了哪些编程语言的规范（提示：有三种）？

第2部分
开发前的准备

工欲善其事，必先利其器。学好小程序的开发，首先要对开发工具有充分的了解。

本章我们对小程序的注册申请、小程序管理后台的使用、开发工具的安装和使用分别做详细的介绍。



本章要点

- 小程序的注册申请和微信认证
- 认识管理后台
- 开发工具编辑模式和调试模式的使用

第3天

小程序的注册申请

3.1 申请微信小程序

申请小程序首先要有一个没注册过微信公众号的邮箱，请牢记邮箱名称和密码，因为我们将来需要它接收微信的官方通知或者完成取回密码等操作。

在微信公众平台官网首页（mp. weixin. qq. com）单击右上角的“立即注册”按钮进入申请流程。如图3-1所示为微信公众平台注册登陆入口。



图3-1 微信公众平台注册登陆入口

进入微信公众平台的注册页，在这个页面单击“账号类型区别”可以查看不同类型账号的区别，我们选择“小程序”选项。

接下来填写注册邮箱和密码，腾讯会发送一份邮件到你的注册邮箱，单击里面的激活链接，小程序的注册和激活就完成了。

邮箱激活后，继续下一步的注册流程，需要选择主体类型、完善主体信息和管理员信息。

申请时一定要慎重填写小程序名称，因为一旦审核通过就不能更改了，服务类目选项决定了我们的小程序在微信搜索引擎的曝光率，所以也要慎重填写，一定要跟小程序的实际功能契合，并且选择尽量多的有关类目，此选项每个月只能修改 1 次。

目前微信小程序只支持企业、政府等机构注册，不支持个人注册。目前包括企业、政府、媒体、其他组织类型账号，必须通过微信认证验证主体身份。认证通过前，小程序的上传发布功能将无法使用，开发者和体验者的账号数量也会受到限制。如图 3-2 所示为选择微信公众平台类型。



图 3-2　选择微信公众平台类型

3.2　微信认证

微信认证的目的是确认开发者提交的小程序主体信息的真实性，需要交纳 300 元的认证费用给腾讯指定的账户，比较方便的方式是通过微信支付扫码提交认证费用。

政府、媒体、其他组织类型账号，必须通过微信认证验证主体身份。企业类型的小程序账号，可以根据需要确定是否申请微信认证，通过微信认证的账号才能使用微信支付、微信卡券等微信提供的比较高级的权限。

微信的审核和认证流程很严格，我们要谨慎对待，认证的名称一旦确定，不易修改，如果需要修改名称，需要我们重新认证并且再次交纳认证费用。

认证入口位于小程序后台的“设置”—“基本设置”—“微信认证”—“详情”一栏。

大部分小程序开发者都是企业类型的用户，所以我们重点说说企业类型的认证方式，一

种是通过腾讯对注册公司的对公账户向腾讯公司打款来验证主体身份的方式，打款信息在提交主体信息后可以查看到。另外一种方式是通过微信认证验证主体身份，需支付 300 元认证费。如果不仅仅是为了尝试小程序的开发而没有上线的计划，建议采用第二种认证方式，因为腾讯小程序如果要上线发布必须通过微信认证，企业认证必须是办理过三证合一的企业营业执照。如图 3-3 所示为微信小程序的微信认证入口。



图 3-3　微信小程序的微信认证入口

微信官方的声明是需要 10 到 15 个工作日才能认证成功的，但是目前只要 1 至 3 个工作日就可以认证成功了，效率很高。

微信认证完成后，我们就可以使用微信小程序管理后台提供的所有功能了。

3.3　小程序绑定微信开放平台账号

如果我们注册了微信开放平台，可以将小程序账号绑定到微信开发者平台，绑定后的小程序可以与账号下的其他移动应用、网站应用或者公众号打通，通过 UnionID 机制让多个小程序应用和公众号之间打通用户账号系统。只要是同一个微信开放平台账号下的移动应用、网站应用和公众账号（包括小程序），用户的 UnionID 是唯一的。换句话说，同一个用户，对同一个微信开放平台下的不同应用，UnionID 是相同的，打通 UnionID 可以提高用户体验的一致性，也可以帮我们更高效地管理用户。

用户的 UnionID 可以通过调用“获取用户信息”接口获取。

微信公众平台绑定小程序的入口在：登录微信开放平台（open. weixin. qq. com）后台的“管理中心”—“公众帐号”—“绑定公众帐号”。如图 3-4 所示。

想要在微信开放平台绑定小程序，首先要有微信平台的开发者资质，开发者资质的认证与小程序微信认证的步骤基本上一样，我们就不单独介绍了。

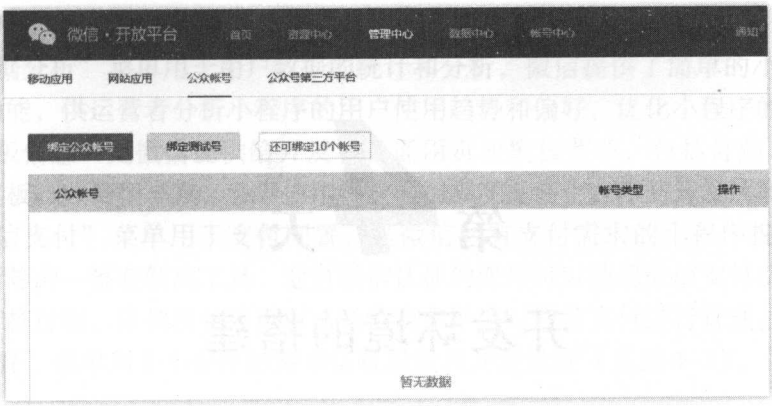


图 3-4　小程序绑定到微信开放平台



课后作业

申请一个微信小程序开发者资格（不需要通过微信认证），为微信小程序的开发做准备。

第4天

开发环境的搭建

4.1 认识小程序管理后台

成功申请后登陆到小程序管理后台后，会进入如图 4-1 所示的界面。

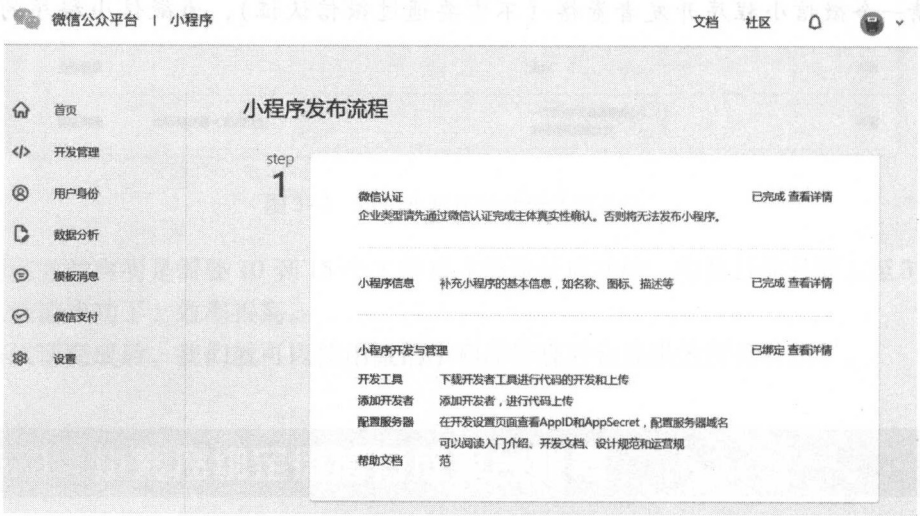


图 4-1　小程序管理后台

界面最顶端是标题栏，包括微信小程序的开发文档、开发者社区入口、通知入口和用户管理。

左侧是导航菜单，共分“首页”、“开发管理”、“用户身份”、“数据分析”、“模板信息”、“微信支付”、“设置”等七个部分。右下是管理后台的主要显示区。

（1）“首页”包括小程序的发布流程信息和版本发布工具，发布流程信息包括微信认证的状态、小程序信息，以及小程序的开发工具下载、开发者账号管理、服务器配置等信息。

（2）“开发管理”菜单包括上线版本的信息、小程序版本审核信息，以及正在开发尚未发布的版本信息。

（3）“用户管理”菜单用于管理员账号的修改、开发者和体验者账号的绑定、解绑。通过微信认证的小程序可以绑定 20 个开发者和 40 个体验者账号，未通过微信认证的小程序可

以绑定 10 个开发者和 20 个体验者账号。

（4）“数据分析”菜单用于用户数据的统计和分析，微信提供了简单的小程序用户数据统计和分析功能，供运营者分析小程序的用户使用趋势和偏好，优化小程序的功能。

（5）“模板信息”是微信提供给开发者的通用页面模板菜单，包括订购、行程、投保、点评等各类模板，数量很丰富，合理使用模板能大幅度提高小程序的开发效率。

（6）“微信支付”菜单用于支付配置，是微信向有支付需求的小程序提供的支付、收款、经营分析等的一整套解决工具，通过微信认证的账号可以使用微信支付功能。微信支付需要独立的审核过程，审核通过后可以在微信商户平台对微信支付进行管理。

（7）“设置”菜单用于小程序的基本信息设置和开发设置（见图 4-2）。

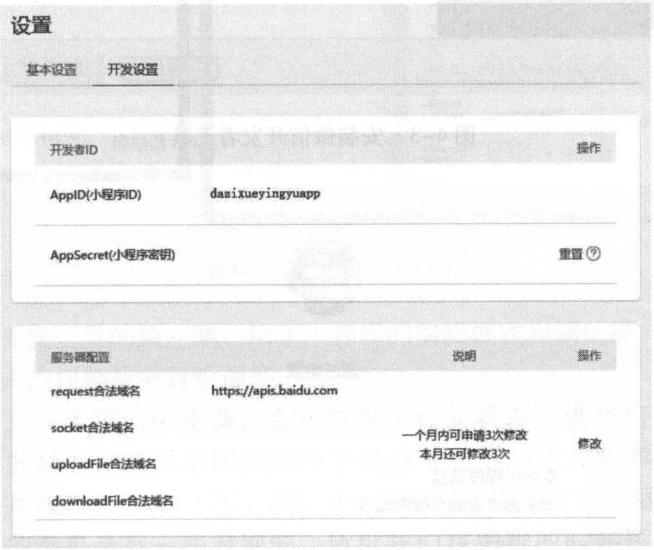


图 4-2　开发设置

4.2　小程序开发环境的搭建

下载小程序开发工具的链接位于微信小程序的管理后台的“首页”－“开发工具”页面下，安装的过程很简单，一路单击“下一步”按钮就可以完成安装了。如图 4-3 所示。

第一次启动开发工具，会要求扫描一个二维码验证开发者身份，我们用绑定了开发者账号的微信号扫描这个二维码，就能启动微信小程序开发工具了。

开发工具启动后首先要进行调试类型的选择，可选类型包括“本地小程序项目”和“公众号网页开发”，如图 4-4 所示。因为我们要做的是小程序的开发，所以选择第一项。进入到“添加项目”选单，在这里我们需要输入小程序的 AppID，前文已有介绍，AppID 在小程序管理后台的“设置”－“开发设置”里获取，接下来我们填写“项目名称”和项目的保存路径，开发工具会完成创建项目和创建项目目录等操作，进入小程序开发工具的主界面。

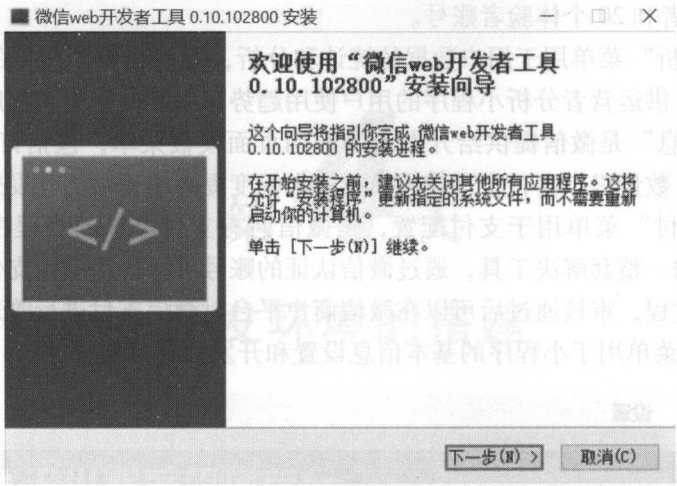


图 4-3　安装微信开发者工具



图 4-4　选择调试类型

4.3　初识小程序开发工具

开发工具主界面分为四个区域：标题菜单栏、导航按钮、模拟器和主操作区。这个布局和 Chrome 开发者工具非常相似。

最顶部的标题菜单栏的三个菜单按钮分别是“设置”、“动作”和“帮助”，“设置”按钮用于网络代理服务器的设置，如果我们是通过代理网址上网的，就需要对这个选项进行设

置，“动作”按钮用于模拟器的页面刷新、前进和后退操作，“帮助”按钮目前只有开发工具的关于信息。如图 4-5 所示。

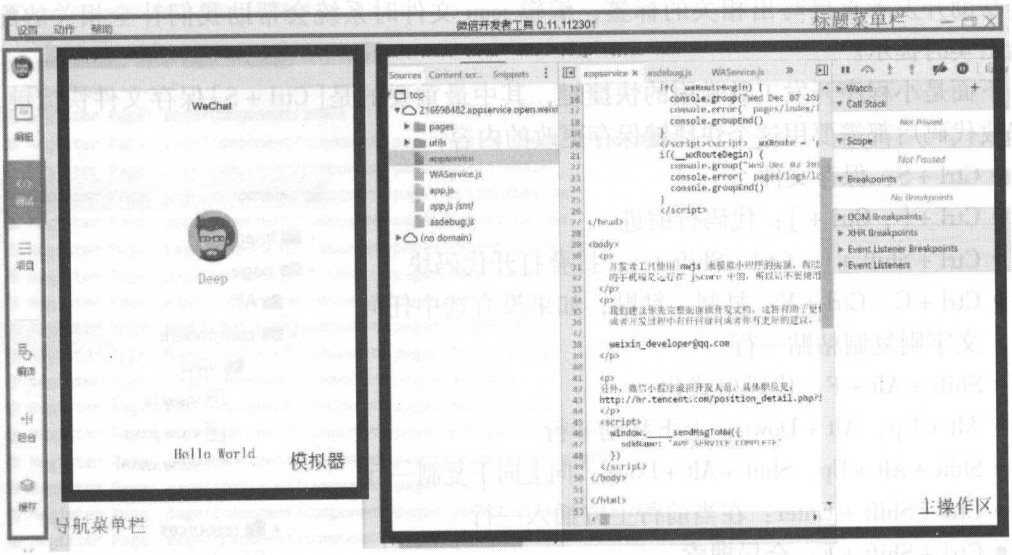


图 4-5　开发工具主界面

导航菜单位于开发工具的最左侧，用于小程序的编辑模式和调试模式的切换、代码的编译、后台和前台模式的切换及项目管理等。

模拟器窗口就是一个页面浏览器，会实时显示页面状态，供我们操作和调试，我们可以在模拟器窗口的最上方选择不同种类的手机型号和不同的分辨率，也可以选择模拟不同的上网方式，系统提供 WiFi、2G、3G、4G 四种网络连接方式。目前开发工具的模拟器和我们真机运行的结果是有一些差别的，所以我们在编程的时候还是要以真机运行的结果为准。

4.4　编辑模式

我们在导航菜单击“编辑”按钮进入编辑模式，可以看到，主操作区会显示小程序的目录树和代码编辑区域，目录树用于小程序的文件管理，我们的代码编写工作需要在代码编辑区域完成。

在“目录树”窗口我们可以方便地查看和管理文件，如图 4-6 所示。单击“目录树”右上角三个点的按钮可以从硬盘打开小程序代码文件、查找文件里的关键词和新建目录或者代码文件。

开发工具目前提供了 wxml、wxss、js、json 共 4 种文件的编辑功能和图片文件的预览，在我们编辑代码时，也可以实时地在模拟器里预览代码编辑的情况。

在编写代码的过程中，开发工具会自动帮助用户保存当前的代码编辑状态，关闭工具或者切换到别的项目并不会丢失已经编辑的文件状态，但只有按下[Ctrl + S]快捷键保存代码后，修改过的代码才能够被重新编译。

同现在大部分成熟的代码编辑器一样，小程序开发工具提供了自动补全功能，编辑 js 文件时系统会帮助我们补全所有的 API，并自动给出相关的注释解释，编辑 wxml 文件时系统会帮助开发者直接写出相关的标签，编辑 json 文件时系统会帮助我们补全相关的配置，并给出实时提示。

下面是小程序开发工具常用的快捷键，其中最重要的是[Ctrl + S]保存文件快捷键，每次修改代码后都需要用这个快捷键保存更改的内容：

- Ctrl + S：保存文件
- Ctrl + [，Ctrl +]：代码行缩进
- Ctrl + Shift + [，Ctrl + Shift +]：折叠打开代码块
- Ctrl + C，Ctrl + V：复制，粘贴，如果没有选中任何文字则复制粘贴一行
- Shift + Alt + F：代码格式化
- Alt + Up，Alt + Down：上下移动一行
- Shift + Alt + Up，Shift + Alt + Down：向上向下复制一行
- Ctrl + Shift + Enter：在当前行上方插入一行
- Ctrl + Shift + F：全局搜索

光标操作快捷键：

- Ctrl + End：移动到文件结尾
- Ctrl + Home：移动到文件开头
- Ctrl + i：选中当前行
- Shift + End：选择从光标到行尾
- Shift + Home：选择从行首到光标处
- Ctrl + Shift + L：选中所有匹配
- Ctrl + D：选中匹配
- Ctrl + U：光标回退

界面操作快捷键：

- Ctrl + \ ：隐藏侧边栏
- Ctrl + m：打开或者隐藏模拟器

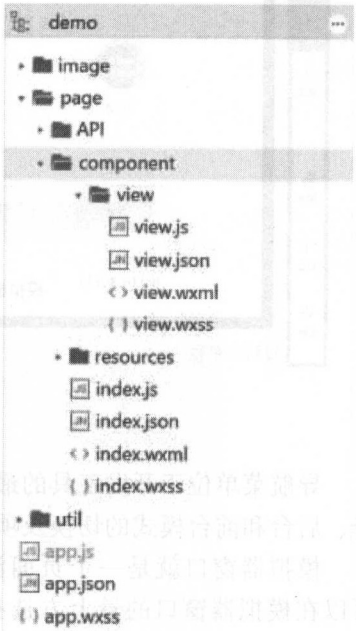


图 4-6　“目录树”窗口

4.5　调试模式

单击导航菜单的“调试”按钮进入调试模式，主操作区会显示调试工具，调试工具有 6 个窗口：“Console”、“Wxml”、“Sources”、“Network”、“AppData”、“Storage”。

“Console”窗口就是小程序的调试控制台，也被叫作终端窗口，用于给开发者输入调试代码和查看调试信息，也用于小程序的错误输出，是我们在小程序调试时最常用的面板。如图 4-7 所示。

“Wxml”窗口用于帮助开发者开发 wxml 文件转化后的界面。类似我们在 Chrome 浏览器窗口查看一个页面时，单击鼠标右键“开发者工具”的“Elements”窗口看到的信息，在这

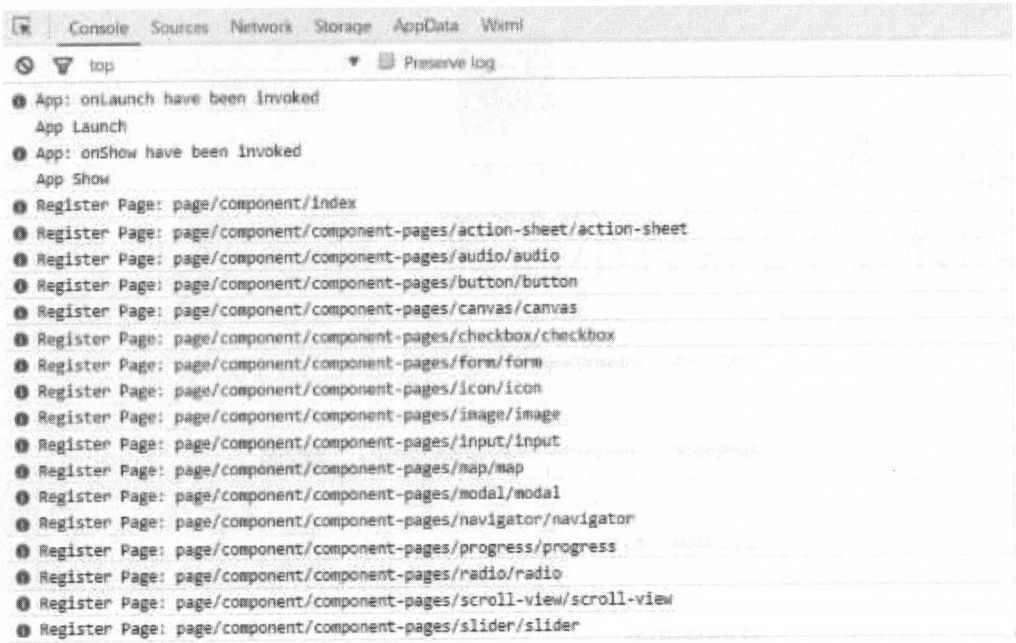


图 4-7　调试面板（当前为 Console 面板）

里可以看到真实的页面结构和结构对应的 wxss 属性，同时可以通过修改 wxss 属性在模拟器中实时看到修改的情况。通过调试模块左上角的选择器，还可以快速找到页面中组件对应的 wxml 代码。关于 wxml 及 wxss 文件，我们会在后面的章节详细介绍。

“Sources”窗口用于显示项目的脚本文件，与浏览器的代码开发不同，微信小程序框架会对脚本文件进行编译，所以在 Sources 面板中开发者看到的文件是经过处理之后的脚本文件，开发者的代码都会被包裹在 define 函数中，并且对于 Page 代码，在尾部会有 require 的主动调用。

“AppData”窗口用于在调试中实时查看和编辑项目数据情况，在此处编辑数据，将会实时反馈到模拟器界面上。

“Storage”窗口是存储器窗口，用于显示项目在使用 wx. setStorage 接口或者 wx. setStorageSync 接口后的数据存储情况。

4.6　项目管理

单击导航菜单栏的“项目”按钮，会进入小程序的项目管理界面，如图 4-8 所示。

项目管理界面用于查看小程序的基础信息和配置信息，也供我们在手机上预览小程序，以及把小程序，代码上传到小程序管理后台。

在项目管理界面单击预览功能，开发工具会自动上传源代码到微信服务器，上传成功后将会显示一个二维码，我们用新版微信扫描二维码就可以在手机上查看和调试自己编辑的小程序了。只有登陆开发工具的微信号可以扫码预览，不支持其他微信号的预览，即使是小程序的管理员。

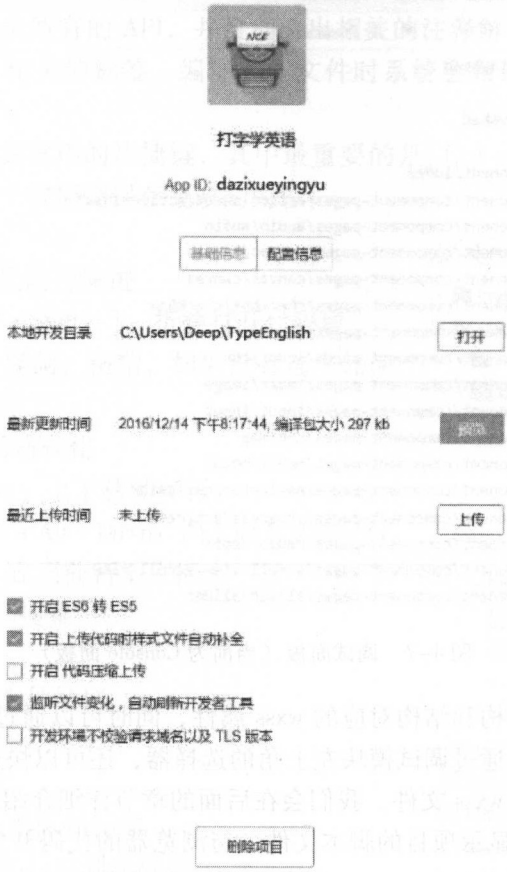


图 4-8　项目管理界面

另外，微信小程序的预览对微信版本也是有要求的，只有 6.3.27 或者以上版本的微信才可以扫码预览。

小程序的管理员可以在项目管理页面单击“上传”按钮上传小程序，代码上传后在小程序的管理后台可以看到提交的版本的状态和进度审核情况。

代码审核通过后，开发者还需要在小程序后台手动单击发布，小程序才会发布到线上。



课后作业

1. 给你申请的微信小程序增加一个开发者账号和一个体验者账号。
2. 说说小程序的代码保存和全局搜索快捷键分别是什么？
3. 小程序的调试终端窗口叫什么名字？可以显示哪些信息？

第3部分
微信小程序的技术框架

一个小程序分为两个部分：视图层（View）和逻辑层（App Service）。视图层负责与小程序使用者的视觉交互，逻辑层负责小程序的数据和逻辑处理，视图层与逻辑层共同构成了小程序的框架。

框架的核心是一个数据绑定系统，也就是视图层和逻辑层的数据通信系统，数据绑定系统能让数据与视图保持同步，当我们在逻辑层修改数据时，视图层相应的被绑定的数据也会更新。

用于小程序视图层描述的编程语言是腾讯自定义规范的 WXML 和 WXSS 语言，逻辑层的代码基于 JavaScript 语言架构，框架在视图层与逻辑层间提供了数据传输的能力和事件系统。



本章要点

- 小程序的目录结构
- 视图层与逻辑层
- JSON 配置文件
- App( ) 函数
- Page( ) 函数
- 小程序的模块化

第5天

框架和配置文件

5.1 小程序的目录结构

腾讯非常贴心地提供了一个微信小程序的 Demo，展示了微信小程序组件和 API 调用，比较全面，本章我们利用这个 Demo 对微信小程序的框架建立起认识。

Demo 的源码可以在小程序管理后台的“文档”—“开发”—“简易教程”—“体验小程序”页面下载，也可以加作者微信号 2264538870 索取。

文件下载后我们解压到一个文件夹里，按照上一章介绍的方法，进入微信小程序开发工具的时候选择 Demo 所在的文件夹就可以打开这个 Demo，进行小程序的编辑和调试。

微信官方提供的所有组件和 API 在 Demo 里都有演示，如视图容器、表单组件等。对我们深入学习和理解微信小程序的组件和 API 很有帮助，我们在本书接下来的学习过程也会用这个 Demo 当作实例来进行。

我们可以在小程序的目录树里直观地了解到小程序的框架和文件结构。

如图 5-1 所示，一个小程序包含一个描述整体程序的主体部分（根目录）和一组或者几组描述页面的 page 文件夹。

小程序的主体部分位于源代码文件夹的根目录下面，通常由三个文件构成：app. js、app. json 和 app. wxss，这三个文件是小程序的架构文件，不能改名。

app. js 是小程序的逻辑代码文件，是逻辑代码的存储容器，文件格式遵守 JavaScript 语言规范。

app. json 是小程序的公共配置文件，用于决定页面文件的路径、窗口内容的配置、设置网络超时时间、设置公共导航栏等。

app. wxss 是小程序的公共样式表文件，兼容 W3C 的 CSS 语法规范。

一个小程序的页面分别由四个文件组成：[pageName]. js、[pageName]. json、[pageName]. wxml、[pageName]. wxss。同一个页面，这四个文件的路径和文件名必须相同，否则小程序在调用和处理文件时会无法识别。

- [pageName]. js 是页面的逻辑文件，负责页面逻辑内容的处理，采用 JavaScript 语言框架。
- [pageName]. json 是页面的配置文件，用来设置页面的窗口内容，遵守 JSON 语法

规范。

- [pageName].wxml 是页面的结构文件，用于页面可视化组件的组织和描述，语法结构类似于 xml，与 html 格式有比较大的差别。
- [pageName].wxss 是页面的样式表文件，兼容 CSS 语法规范。

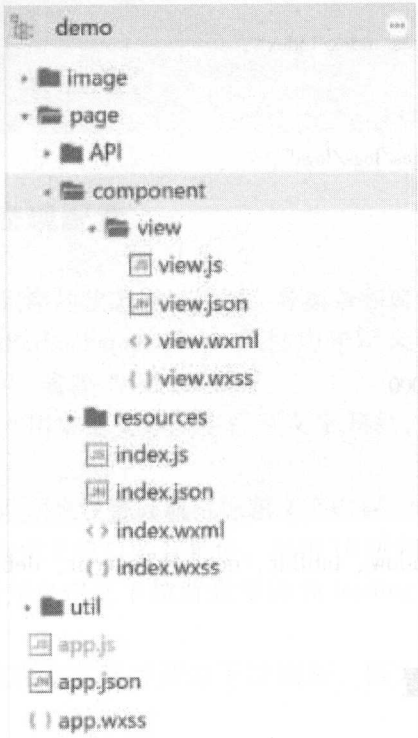


图 5-1　小程序的目录树

5.2　JSON - 小程序的配置文件

小程序的配置文件以 .json 为扩展名，文件格式遵守 JSON 语法规范，分为小程序的全局配置文件 app.json 和页面的配置文件 pageName.json。

项目文件夹根目录的 app.json 是小程序的全局配置文件，决定小程序所有可用页面的路径、窗口表现、小程序的网络超时时间和公共导航栏按钮的设置等。

例如，下面这个 app.json 文件，包含了所有 json 文件可配置选项的简单配置内容。

```
{
  "pages":[
    "pages/index/index",
    "pages/logs/index"
  ],
```

```
  "window":{
    "navigationBarTitleText":"打字学英语"
  },
  "tabBar":{
    "list":[{
      "pagePath":"pages/index/index",
      "text":"课程"
    },{
      "pagePath":"pages/logs/logs",
      "text":"日志"
    }]
  },
  "networkTimeout":{
    "request":10000,
    "downloadFile":10000
  },
  "debug":true
}
```

该文件定义了 pages、window、tabBar、networkTimeout、debug 五个选项，我们分别做详细介绍。

5.2.1　pages – 页面配置

pages 选项用来指定小程序由哪些前端页面组成，它的设置对象是一个由多个字符串构成的数组，每一个字符串都对应页面的路径和文件名信息，需要特别注意的是这个数组的第一项是小程序启动的初始页面信息。

如果我们对小程序的页面做了增加或者删除的操作，也需要修改 pages 数组的内容，否则修改后的页面将无法加载，小程序的编译也会报错。

数组字符串的文件名不需要写文件后缀，小程序框架会根据语法和功能需求，自动寻找路径下的“. json”、“. js”、“. wxml”、“. wxss”的四个文件来处理。

例如，如果我们的项目文件夹结构是这样的：

```
pages/
pages/TypeEng/index. wxml
pages/ TypeEng /index. js
pages/ TypeEng /index. wxss
pages/logs/logs. wxml
pages/logs/logs. js
app. js
app. json
app. wxss
```



我们 app. json 中 pages 选项就需要定义成如下的样式：

```
{
  "pages":[
"pages/ TypeEng /index"
    "pages/logs/logs"
  ]
}
```

5.2.2　window – 窗口表现配置

window 选项用来设置小程序的状态栏、标题、导航条和窗口背景颜色等窗口表现内容。

window 选项的 navigationBarBackgroundColor 属性用来定义导航栏的背景颜色，采用 16 进制颜色格式，如“#000000”或者“#FFFFFF”。

navigationBarTextStyle 属性用来定义导航栏标题文字颜色，只支持“black”和“white”两个数值，非黑即白。

navigationBarTitleText 属性用来设置导航栏标题文字内容。

backgroundColor 属性用来定义窗口的背景色，采用 16 进制颜色格式。

backgroundTextStyle 属性用来定义下拉背景字体和 loading 图的样式，仅支持 dark/light 两个数值。

enablePullDownRefresh 属性定义是否开启下拉刷新，值为“true”时开启，为“false”时禁止开启。

我们可以把 demo 的 app. json 的 window 对象改成如下的配置，通过在模拟器里显示效果的改变来了解各项属性和参数带来的效果。

```
{
  "window":{
"navigationBarBackgroundColor":"#ffffff",
    "navigationBarTextStyle":"black",
    "navigationBarTitleText":"打字学英语 app 演示",
    "backgroundColor":"#eeeeee",
    "backgroundTextStyle":"light"
  }
}
```

5.2.3　tabBar – 导航栏配置

小程序用 tabBar 选项配置公共导航栏，tabBar 的对象是一个数组，这个数组只能配置 2 到 5 个元素，也就是说小程序的导航栏只支持设置 2 到 5 个导航按钮（tab），这是因为微信认为只有一个导航栏按钮会失去“导航”的意义，而多于五个导航按钮会造成导航栏过于

拥挤，视觉效果过于混乱，导航按钮的位置按数组的顺序排列，第一个数组的元素对应导航最左边的按钮，依次排列。

tabBar 对象可以配置六种属性，它们分别是 color、selectedColor、backgroundColor、borderStyle、list 及 position。

（1）color 和 selectedColor 属性用于设置导航栏文字的默认颜色和被选中时表现的颜色，小程序颜色的定义方法我们会在 WXSS 的学习中详细讲解。

（2）backgroundColor 用于设置导航栏的背景色。

（3）borderStyle 用于设置导航栏的上边框颜色，只支持 black 和 white 两种。

（4）position 用于设置导航栏在页面的位置，在顶部或者底部，分别用数值 top 和 bottom 定义。

（5）list 属性用来配置导航栏导航按钮，它的格式是数组，每一个数组定义一个按钮的配置，数组中的每一项都是一个对象，分别用来定义按钮被单击时跳转到的页面路径、按钮文字、按钮默认显示的和选中时显示的图标路径等属性，其中：

- pagePath 属性用于设置导航按钮被单击时跳转到的页面路径；
- text 属性用来定义按钮的文字；
- iconPath 被用来定义默认按钮图片的路径，selectedIconPath 用来定义按钮被选中时显示的图片路径，需要注意的是导航栏按钮 icon 图片的大小不能超过 40kb，只有底部显示的导航栏可以显示设置的按钮图片，顶部导航栏设置这个参数无效，图片不会被显示出来。

5.2.4　networkTimeout – 通信超时事件配置

networkTimeout 对象用于设置小程序在网络通信时的超时时间，单位是毫秒，网络环境比较差的情况下超时事件太短会造成小程序频繁报告通信故障错误。

networkTimeout 对象有五个可以设置的属性：

- request 用来设置 wx. request 的超时时间，单位是毫秒，wx. request 是用来向服务器发起 https 网络请求的函数。例如，我们需要让小程序判断这样一个操作，如果 10 秒钟没有获得响应就显示网络出错警告，我们就须要设置 request 的值为 10000；
- connectSocket 用于配置 wx. connectSocket 的超时时间，wx. connectSocket 是小程序用来创建 WebSocket 链接的函数；
- uploadFile 用于配置 wx. uploadFile 的超时时间，wx. uploadFile 是用来将本地资源上传到小程序后台服务器的函数；
- downloadFile 用于配置 wx. downloadFile 的超时时间，wx. downloadFile 是小程序用来下载文件资源到本地的函数。

上面提到的函数可以在本书附录的 API 一节获得详细的资料。

5.2.5　debug – 调试模式配置

小程序用 Debug 配置来开启小程序开发工具 debug 模式，debug 模式开启后，在开发者



工具的 Console 控制台面板，调试信息会以 info 的形式陈列出来，这些信息有 Page 的注册、页面路由、数据更新、事件触发等，方便我们快速定位一些常见的调试问题。

每一个小程序页面分别可以用对应页面路径下的 .json 文件来配置本页面的窗口展示效果。

因为页面的配置不会涉及公共导航栏、通信超时事件等属性，所以页面的配置比 app.json 全局配置要简单得多，只须设置 window 配置项的内容，加载页面时，page.json 的 window 配置项会覆盖 app.json 中相同配置项的内容。

页面的 .json 文件只设置 window 配置项的属性，所以我们不须声明 window 这个键。例如，下面的这个 .json 页面配置文件内容，也是符合小程序配置文件的语法规范的：

```
//index.js
{
  "navigationBarBackgroundColor":"#ffffff",
  "navigationBarTextStyle":"black",
  "navigationBarTitleText":"打字学英语 app 演示",
  "backgroundColor":"#eeeeee",
  "backgroundTextStyle":"light"
}
```



课后作业

1. 用微信小程序开发工具新建一个空文件夹作为小程序的根目录，并新建根目录下的小程序主体架构文件。

2. 在上题结果的基础上，新建一个名为“打字学英语”的页面目录，新建目录下的页面架构文件。

3. 修改根目录下 .json 的文件内容，配置上题新建的页面并设计两个在屏幕底端位置的导航栏按钮，分别定名为“选课”和“练习”。

第6天

逻辑层（AppService）

小程序开发框架的逻辑层包括根目录和页面文件夹下的.js文件，文件格式遵循JavaScript语言规范，逻辑层负责小程序的数据处理，逻辑层从视图层接收数据，处理后再把结果返回给视图层。

小程序使用了App和Page方法进行程序和页面的注册，使用getApp方法来获取App()实例。

在面向对象编程中，“实例”（instance）是个很重要的概念，实例是基于“类”的概念来说的。“类”是模拟我们显示世界的“种类”产生的概念，是一个框架，一种规范。例如，“狮子”就是一个类，而一只具体的狮子，如它叫“Eric”，就是“狮子”这个类的一个实例，所以类是静态的，不占内存的，而“实例”在内存中有一个位置来存放。

在面向对象编程语言中，类的实例是通过“对象”来实现的，而JavaScript的对象是基于函数的，如小程序本身就是通过一个函数App()实现的，小程序的页面是通过Page()函数实现的，所以我们可以认为，小程序的“实例”也就是构成小程序的各个函数。

关于类、对象、函数等概念我们会在JavaScript进阶一章做更加详细的讲解。

针对微信用户数据、扫一扫、微信支付等微信独有的能力，微信提供了丰富的API函数给开发者调用。

小程序的每个页面都有独立的作用域，并且我们可以给页面文件做模块化处理。

由于小程序框架是在微信浏览器运行的，并非IE、Chrome等公用浏览器，所以小程序的.js文件对JavaScript的一些有关Web的能力做了优化，如DOM（文档对象）、window等公用浏览器常用对象无法在小程序中使用。

开发者在开发工具编写的所有逻辑层代码最终将会打包成一份JavaScript文件，在小程序启动的时候这个文件被下载下来运行，并且有离线执行的能力，直到小程序销毁。

这种机制类似于W3C的ServiceWorker架构，代码可以做到离线使用，所以微信把逻辑层称为AppService。

6.1 小程序的初始化

小程序的注册和初始化是通过App()函数进行的，App()注册小程序时接受一个object参数，指定小程序的生命周期。



一个对象的生命周期，也就是从对象创建运行、提供服务、暂时终止服务、到最终被销毁的整个过程。

表 6-1 是 App( ) 函数中有关微信小程序生命周期的 object 参数说明。

表 6-1　有关微信小程序生命周期的 object 参数说明

属　性	类　型	描　述	触发时机
onLaunch	Function	生命周期函数—监听小程序初始化	当小程序初始化完成时，会触发 onLaunch（全局只触发一次）
onShow	Function	生命周期函数—监听小程序显示	当小程序启动，或从后台进入前台显示，会触发 onShow
onHide	Function	生命周期函数—监听小程序隐藏	当小程序从前台进入后台，会触发 onHide
其他	Any	开发者可以添加任意的函数或数据到 Object参数中，用 this 可以访问	

onLaunch 属性作用于小程序的初始化阶段，小程序的一个完整生命周期里 onLaunch 属性只会触发一次。

小程序启动或者从后台转入前台运行时会触发 onShow 属性，当小程序从后台切换到前台运行时会触发 onHide 属性。

我们来说说程序的后台和前台运行的概念，当用户单击小程序的关闭按钮，或者按了手机的 Home 键离开微信，或者接电话离开了小程序页面时，小程序并没有被直接销毁，而是进入了后台运行状态，当用户再次进入微信或者打开小程序时，小程序会恢复到前台运行状态，而不需要被重新启动加载。只有当小程序进入后台一段时间或者系统资源占用过高时，才会被系统真正销毁，这个销毁的时间是由手机操作系统（OS）决定的。

例如，下面的 App( ) 函数示例，在小程序初始化时指定了上述的三个生命周期函数和一个全局变量 globalData：

```
//app.js
App({
  onLaunch:function() {
    //当小程序初始化完成时,触发这个函数.
  },
  onShow:function() {
      //当小程序启动,或从后台进入前台显示,触发这个函数.
  },
  onHide:function() {
      //当小程序从前台进入后台,触发这个函数.
  },
 appName:'打字学英语演示'
})
```

小程序框架提供了一个全局的 getApp( ) 函数，可以获取到小程序实例。例如，下面的例子，就是在 index.js 执行时引用了 App( ) 实例：

```
//index. js
var appInformation = getApp( )
console. log( appInformation. appName)          // 打字学英语演示
```

需要注意的是，App( )必须且只能在根目录的 app. js 文件中注册，而且只能注册一个。在 App( )函数内需要拿到 App 实例时，只要使用 this 关键字就可以达到目的，不需要调用 getApp( )。

通过 getApp( )获取实例之后，不能私自调用生命周期函数，否则会出现错误。

6.2　页面注册

我们在程序里需要使用一个页面之前，需要先用 Page( )函数注册这个页面。Page( )函数接受一个 object 参数，用于指定页面的初始数据、生命周期函数、事件处理函数等。

表 6-2 是 Page( )函数的 object 参数说明。

表 6-2　Page( )函数的 object 参数说明

属　性	类　型	描　述
data	Object	页面的初始数据
onLoad	Function	生命周期函数—监听页面加载
onReady	Function	生命周期函数—监听页面初次渲染完成
onShow	Function	生命周期函数—监听页面显示
onHide	Function	生命周期函数—监听页面隐藏
onUnload	Function	生命周期函数—监听页面卸载
onPullDownRefresh	Function	页面相关事件处理函数—监听用户下拉动作
onReachBottom	Function	页面上拉触底事件的处理函数
其他	Any	开发者可以添加任意的函数或数据到 object 参数中，在页面的函数中用 this 可以访问

下面是一个页面的 index. js 代码示例，用 Page( )函数完成了页面的初始化、声明周期的定义等：

```
//index. js
Page( {
  data: {
text:"你好 . "
  },
  onLoad:function( options) {
    //页面初始化 options 为页面跳转所带来的参数 .
  },
```



```
  onReady:function() {
    //页面渲染完成.
  },
  onShow:function() {
    // 页面显示
  },
  onHide:function() {
    // 页面隐藏.
  },
  onUnload:function() {
    //页面关闭.
  },
  onPullDownRefresh:function() {
    //. 下拉刷新
  },
  onReachBottom:function() {
    //上拉加载.
  },
  //事件处理.
click:function() {
    this. setData({
text:'你好,我是打字学英语.'
    })
  },
informationData:{
    hi:'打字学英语'
  }
})
```

初始化数据是对于这个页面的第一次渲染，data 会以 JSON 语法的形式由逻辑层传至视图层，所以 data 数据必须是 JSON 语法支持的格式，如字符串、数值、布尔值、对象、数组等。

视图层通过 WXML 进行数据绑定，我们会在后面的章节详细讲述关于数据绑定方面的内容。

下面是同一页面下的 wxml 文件数据绑定代码：

```
<!--index. wxml-->
<view>{{text}}</view>
<view>{{array[0]. msg}}</view>
Page({
  data:{
text:'打字学英语',
```

```
    array:[{msg:'1'},{msg:'2'}]
  }
})
```

Page()函数可以调用 onLoad、onShow、onReady、onHide、onUnload 五个生命周期函数：

- onLoad 只在页面加载时调用一次，接收页面参数可以获取到 wx. navigateTo 和 wx. redirectTo 事件及 <navigator/> 中的 query 事件；
- onShow 在页面显示时调用，每次打开页面都会调用一次；
- onReady 参数在页面初次渲染完成时调用，每一个页面只会调用一次，代表页面已经准备好，可以和视图层进行数据交互，对界面的设置如 wx. setNavigationBarTitle 函数等可以在 onReady 之后设置；
- onHide 参数在页面隐藏时调用，即当页面进行 navigateTo 页面导航或者用户通过操作底部导航按钮切换到其他页面时被调用；
- onUnload 参数在页面卸载时调用，即当页面执行 redirectTo 或 navigateBack 操作时调用。

Page()函数用 onPullDownRefresh 属性来监听用户下拉刷新事件，但事先需要在 app. json 的 window 配置中开启 enablePullDownRefresh 选项。当处理完数据刷新事件后，可以用 wx. stopPullDownRefresh()函数停止页面的下拉刷新。

除了初始化数据和生命周期函数，Page()中还可以定义事件处理函数。我们可以在视图层组件中加入事件绑定，当事件被触发时，Page()函数中定义的事件处理函数就会被调用，以上关于事件的概念和处理方式，我们会在后面的学习中详细了解。

下面是页面定义事件处理函数的示例代码：

```
<!--index.wxml-->
<view bindtap="viewTap">click me</view>

//index.js
Page({
click:function(){
    console.log('这是打字学英语 app 的演示页面')
  }
})
```

用来从逻辑层发送数据到视图层的函数是 setData，setData 被调用时对应的 this. data 值会被刷新。

直接修改 this. data 值的方法是无效的，不但不能改变页面的状态，还会造成数据的不一致。

setData()被调用时会接受一个对象，以 key、value 对的形式将 this. data 中 key 的对应值改变成 value。其中的 key 可以以数据路径的形式给出，如 array[2].message，a. b. c. d 的形式，并且不需要在 this. data 中预先定义。

下面是示例代码：

```
<!--index.wxml-->
<view>{{text}}</view>
<button bindtap="shiftText">改变数据</button>
<view>{{array[0].text}}</view>
<button bindtap="changeArray">改变数组数据</button>
<view>{{object.text}}</view>
<button bindtap="changeObject">改变对象数据</button>
<view>{{newField.text}}</view>
<button bindtap="addNewField">添加新数据</button>

//index.js
Page({
  data:{
text:'打字学英语',
    array:[{text:'打字学英语'}],
    object:{
text:'打字学英语'
    }
  },
shiftText:function() {
    this.setData({
text:'改变这个数据'
    })
  },
  changeArray:function() {
    //你可以用这种方式来修改一个动态数据路径
    this.setData({
      'array[0].text':'改变这个数据'
    })
  },
changeObject:function(){
    this.setData({
      'object.text':'改变这个数据'
    });
  },
  addNewField:function() {
    this.setData({
      'newField.text':'新增数据'
    })
  }
})
```

getCurrentPages( )函数用来获取当前页面栈的实例，按栈的顺序给出，形式是一个数组，数组的第一个元素是首页，最后一个元素是当前页面。需要注意的是页面栈是不能直接修改数据的，否则会造成页面的状态和路由错误。

栈也就是堆栈，是一个数据类型和存储的概念，堆栈中的物体具有一个特性：最后一个放入堆栈中的物体总是被最先拿出来，这个特性被称为后进先出。堆栈中定义了一些操作。两个最重要的是入栈（push）和出栈（pop）。入栈操作会在堆栈的顶部加入一个元素。出栈操作相反，执行后会在堆栈顶部移除一个元素，同时堆栈的大小减一。

小程序框架以堆栈的形式维护页面，遵守后进先出的原则，页面切换时，页面栈会有如表 6-3 所示。

表 6-3　页面切换时页面栈的表现

页面路由方式	页面栈表现
初始化	新页面入栈
打开新页面	新页面入栈
页面重定向	当前页面出栈，新页面入栈
页面返回	页面不断出栈，直到目标返回页，新页面入栈
Tab 切换	当前页面出栈，新页面入栈

6.3　页面的路由

页面的路由是指从一个页面跳转到另一个页面的过程，在小程序中所有页面的路由都由框架管理，路由的触发方式和页面生命周期函数如表 6-4 所示。

表 6-4　路由的触发方式和页面生命周期函数

路由方式	触发时机	路由后页面	路由前页面
初始化	小程序打开的第一个页面	onLoad，onShow	
打开新页面	调用 APIwx. navigateTo 或使用组件 < navigator/ >	onLoad，onShow	onHide
页面重定向	调用 APIwx. redirectTo 或使用组件 < navigator/ >	onLoad，onShow	onUnload
页面返回	调用 APIwx. navigateBack 或用户按左上角返回按钮	onShow	onUnload（多层页面返回每个页面都会按顺序触发 onUnload）
Tab 切换	多 Tab 模式下用户切换 Tab	第一次打开 onLoad，onShow；否则 onShow	onHide

6.4　逻辑层的模块化

JavaScript 文件中声明的变量和函数只在这个文件中有效，所以我们可以在不同的文件

中声明相同名字的变量和函数，不会互相影响。

调用全局函数 getApp()可以获取小程序的全局应用实例，如果需要全局有效的数据我们可以在 App()中定义，例如：

```
// app.js
App({
appName:1
})

//index.js
//本地的只能用在文件 a.js。.
var localValue ='a'
//获取这个应用实例
var app = getApp()
//获取全局数据并更改它
app.appName ++

// b.js
//你可以重新在本地定义文件 b.js,不受本地的 a.js 文件干扰。
var localValue ='b'
console.log(getApp().appName)
```

为了提高代码的可读性和提高代码效率，我们可以利用小程序的模块化特性，模块化就是将公共的或者需要复用的代码从程序里抽取出来，打包成单独的.js 文件，提供某种特定的功能，这就是程序的模块。小程序的 JavaScript 代码模块只能通过 module.exports 和 exports 方法对外暴露接口。

以下示例中的 modular.js 就是一个小程序模块：

```
//mudular.js
function sayHello(name) {
  console.log('Hello${name}!')
}
function sayGoodbye(name) {
  console.log('Goodbye${name}!')
}

module.exports.sayHello = sayHello
exports.sayGoodbye = sayGoodbye
```

我们在 index.js 中使用 require（path）方法将以上 js 模块的公共代码导入：

```
//index.js
var modular = require('modular.js')
Page({
```

```
    hello:function() {
modular. sayHello('打字学英语')
    },
    goodbye:function() {
modular. sayGoodbye('打字学英语')
    }
})
```



课后作业

1. 新建一个小程序，在 App()函数内定义 onLaunch、onShow 和 onHide 函数，并定义一个全局变量 appData，变量的值是“打字学英语”。

2. 在上题的基础上，新建一个页面，创建 index. js，完善生命周期函数并定义一个变量 text，值是“打一课，会一课!”。

3. 在页面目录里新建一个模块 module. js，用 console. log 方法输出一段内容是“Hello English.”的文字，再用页面目录的 index. js 调用这个模块。

第4部分
WXML语言与视觉组件

小程序的视图层就是我们在手机上打开小程序看到的并且可以操作的视觉页面，视图层负责页面视觉元素的组织和与用户的交互，视图层与逻辑层之间依靠数据传输和事件系统连接起来。

视图层文件分为两个部分，.wxml文件和.wxss文件，其中.wxml文件负责页面结构的描述，基于XML语言规范，也就是微信所说的WXML语言；.wxss文件负责组件的视觉样式描述，基于CSS语言规范，也就是微信所说的WXSS语言。

同时微信也提供了丰富的视觉组件来进行页面的内容组织和展示，视觉组件类似HTML语言的标签，是构成小程序页面的基本用户界面元素。



本章要点

- WXML语言语法规范
- 视觉组件
- 数据绑定
- 逻辑渲染
- 模板
- 事件

第7天

WXML 语言和数据绑定

7.1 WXML 语法规范

WXML 是 WeiXin Markup Language 的简写，翻译过来就是“微信标记语言”，是微信规范标准的一种用于描述页面布局的标签语言，它基于 XML 语言规范，WXML 结合微信提供的基础组件和事件系统来构建页面结构。

微信的视觉组件类似 HTML 语言里的标签，事件系统是视图层与逻辑层的通信方式，事件可以将用户的操作从视图层反馈到逻辑层，也把逻辑处理的结果反映到视图页面上，让用户浏览。

WXML 继承了 XML 语言的语法规范，XML 是可扩展标签语言（Extensible Markup Language）的缩写，在 XML 语言里，标签（markup）是最关键的部分，我们可以在里面创建内容，然后使用限定标签标记它，从而使每个视觉元素和模块成为可识别、可分类的信息。

WXML 文件由内容和标签组成，用标签包围内容的方式将大部分内容包含在元素中，WXML 的标签也就是微信所说的视觉组建，也被称作 WXML 元素。

例如，我们要创建一段 WXML 文本，文本内容是“打字学英语 app，让你轻松过英语六级。”为了让 WXML 明白这是一段文本，我们要将这个文本包含到 text 组件中，分别在文本的首末两端添加文本的开始和结束标签。组件的开始标签就是把组件名放到尖括号中（< >）：<text>。再输入文本“打字学英语 app，让你轻松过英语六级。”在文本的后面加入结束标签：将元素名放在尖括号里面，然后在组件名前面加上一个代表组件结束的斜杠（/），就像这样：</text>。我们就得到了用于达成目的的这样一段 WXML 代码：

```
<text>打字学英语 app，让你轻松过英语六级。</text>
```

总结 WXML 语言的语法规范如下：

- WXML 文档里同样属性的开始标签和结束标签必须成对出现，结束标签的组件名称前要有一个斜杠（/）；
- 也可以用 <template.../> 这种标签形式，叫作空元素标签；
- WXML 标签内的属性要严格区分大小写，如 <text> 和 <Text> 是两个不同的标签；
- WXML 标签的属性值必须用双引号括起来，如 <template is="footer" />；
- WXML 的注释语句形式是 <!--注释语句内容-->，注释内容可以是一行或者多行



文本，例如：

```
<!--
姓名:老王
年龄:28
微信 2264538870;
-->
```

7.2　视觉组件

小程序的视觉组件也就是 HTML 文件中的“标签”，组件是 WXML 语言最基本的视觉单位，小程序框架为开发者提供了一系列的基础组件，让开发者可以通过组合这些基础组件快速地实现想要的功能。

作为小程序视图层的基本组成单元，组件自带一些功能，并且采用了与微信风格相一致的样式，一个组件通常包括开始标签和结束标签，属性位于两个标签之内，用来修饰组件，例如：

```
<!-- index. wxml -- >
<view >
<text >这是打字学英语 APP 的一个页面 </text >
</view >
```

需要注意的是，所有的组件名与属性都是小写的，以连字符（-）连接，组件的描述语法有如下规则：

- 关键词用尖括号包围起来，如 <view >；
- 关键词通常是成对出现的，如 <text > 和 </text >；
- 组件标签对中的第一个标签是开始标签，第二个标签是结束标签，开始和结束标签也被称为开放标签和闭合标签；
- 组件标签属性中赋值的语法形式是“组件名 + 属性名 + 属性值”的形式，属性名与值之间用赋值符号连接，值用双引号括起来，如 <view id = "outter" bindtap = "handleTap1" >。

组件属性的类型有如表 7 - 1 所示的七种。

表 7-1　组件属性的类型

类　型	描　述	注　解
Boolean	布尔值	组件写上该属性，不管该属性等于什么，其值都为 true，只有组件上没有写该属性时，属性值才为 false。如果属性值为变量，变量的值会被转换为 Boolean 类型
Number	数字	1，2.5
String	字符串	"string"

续表

类　型	描　述	注　解
Array	数组	[1,"string"]
Object	对象	{ key:value }
EventHandler	事件处理函数名	"handlerName" 是 Page 中定义的事件处理函数名
Any	任意属性	

所有组件都有的属性我们称为组件的基本属性，如组件的 id 名、样式等，组件的基本属性如表 7-2 所示。

表 7-2　组件的基本属性

属 性 名	类　型	描　述	注　解
id	String	组件的唯一标示	保持整个页面唯一
class	String	组件的样式类	在对应的 WXSS 中定义的样式类
style	String	组件的内联样式	可以动态设置的内联样式
hidden	Boolean	组件是否显示	所有组件默认显示
data - *	Any	自定义属性	组件上触发事件时，会发送给事件处理函数
bind * /catch *	EventHandler	组件的事件	详见事件

每种组件都有根据它自身特点定义的属性，也就是组件的特殊属性，本书的附录一是我们收集的小程序基础组件的属性和例程，大家可以作为工具书来检索使用。

7.3　组件的数据绑定

数据绑定是视觉层数据与逻辑层数据通信的方法，也就是将一个用户界面元素（组件）的属性绑定到一个逻辑文件的对象实例上的某个属性的方法。从视觉组件的角度来看，“数据绑定”是一种把程序数据绑定到用户界面元素的机制。

例如，我们在 JavaScript 程序里有一个 Customer 类型的实例，那么他就可以把 Customer 的“Name”属性绑定到页面上一个 view 组件的“item”属性上。当这两个属性值被绑定了之后，对 view 的 item 属性的更改将“传播”到 Customer 的 Name 属性上，而对 Customer 的 Name 属性的更改同样会“传播”到 view 的 item 属性上。

WXML 中的动态数据来自对应页面逻辑层（.js 文件）的 data 数据。

7.3.1　数据的简单绑定

数据绑定使用一种叫作 Mustache 语法的形式，就是将属性变量用双层花括号（{{}}）括起来的形式进行绑定，可以作用于内容、组件属性、控制属性、关键字的绑定。

（1）示例 1，文本内容的绑定：

```
<!--index.wxml-->
<view>{{text}}</view>

//index.js
Page({
  data:{
text:'Hello 打字学英语!'
  }
})
```

(2) 示例 2，组件属性的绑定。组件属性需要放置在双引号内：

```
<!--index.wxml-->
<view id="item-{{say}}"></view>

//index.js
Page({
  data:{
say:"hello,打字学英语";
  }
})
```

(3) 示例 3，控制属性的绑定。控制属性需要放置在双引号内：

```
<!--index.wxml-->
<view wx:if="{{myKey}}"></view>

//index.js
Page({
  data:{
myKey:true
  }
})
```

(4) 示例 4，关键字的绑定。关键字需要放置在双引号内：

```
<!--index.wxml-->
<checkbox checked="{{true}}"></checkbox>
```

要注意这个例子中 checked="{{true}}"与 checked="true"的表示方法区别很大，前者判断的是绑定的数据，后者是一个内容为“true”的字符串。

7.3.2　绑定时的运算

数据绑定时，可以在双花括号（{{}}）内进行简单的运算，支持下列几种方式：

（1）示例 1，三元运算，即执行一个对条件表达式 type 的判断，结果为真时执行第一个表达式，否则执行第二个：

```
<!--index.wxml-->
<view hidden="{{type ? true :false}}"> Hidden </view>
```

（2）示例 2，算数运算：

```
<!--index.wxml-->
<view> {{a+b}} + {{c}} +d </view>

//index.js
Page({
  data:{
    a:1,
    b:2,
    c:3
  }
})
```

本例在数据绑定时进行了数学运算，运算后 view 中的内容为 3 +3 + d。

（3）示例 3，逻辑判断：

```
<!--index.wxml-->
<view wx:if="{{newLen > 10}}"> </view>
```

（4）示例 4，字符串运算：

```
<!--index.wxml-->
<view>{{"hello" +name}} </view>

//index.js
Page({
  data:{
    name:'微信:2264538870'
  }
})
```

（5）示例 5，数据路径运算：

```
<!--index.wxml-->
<view>{{object.key}} {{array[0]}} </view>
```

```
//index.js
Page({
  data:{
    object:{
      key:'Hello'
    },
    array:['myWX:2264538870']
  }
})
```

7.3.3　绑定时的数据组合

数据绑定时，也可以在 Mustache 表达式内直接进行数据的组合，构成新的数组或者对象：

(1) 示例 1，组合成数组：

```
<!--index.wxml-->
<view wx:for="{{[zero,1,2,3,4]}}">{{item}}</view>

//index.js
Page({
  data:{
    zero:0
  }
})
```

绑定后，最终组合成了数组 [0, 1, 2, 3, 4]。

(2) 示例 2，组合成对象：

```
<!--index.wxml-->
<template is="wordGroup" data="{{for:a,bar:b}}"></template>

//index.js
Page({
  data:{
    a:1,
    b:2
  }
})
```

绑定后组合成的对象是 {for: 1, bar: 2}

也可以用扩展运算符（...）将一个对象展开：

```
<!-- index.wxml -->
<template is="wordGroup" data="{{...obj1,...obj2,e:5}}"></template>

//index.js
Page({
  data:{
    obj1:{
      a:1,
      b:2
    },
    obj2:{
      c:3,
      d:4
    }
  }
})
```

最终组合成的对象是 {a:1,b:2,c:3,d:4,e:5}。

如果对象的 key 和 value 相同，也可以间接地表达成如下形式：

```
<!-- index.wxml -->
<template is="objectCombine" data="{{foo,bar}}"></template>

//index.js
Page({
  data:{
    foo:'my-first',
    bar:'my-last'
  }
})
```

最终组合成的对象是 {foo: 'my-first',bar: 'my-last'}。

上述方式可以随意组合，但是如果变量名相同，后边的数据会覆盖前面的，例如：

```
<!-- index.wxml -->
<template is="wordGroup" data="{{...flg1,...flg2,a,c:6}}"></template>

//index.js
Page({
  data:{
```

```
    flg1:{
        a:1,
        b:2
      },
    flg2:{
        b:3,
        c:4
      },
      a:5
    }
})
```

最终组合成的对象是 {a: 5, b: 3, c: 6}



课后作业

1. 新建一个小程序并新建一个页面，在页面里添加一个 view 标签，在标签里用我们学到的数据绑定知识写一个名为 message 的变量，在同名的 js 文件中新建一个 Page 函数，在函数的 data 里，写入 message，message 对应的值是“Hello，我是打字学英语 app”。

2. 用小程序开发工具将我们在本节中学到的知识点和案例实现一遍。

第8天

逻辑渲染语句

8.1 wx:if 条件渲染语句

渲染，也就是我们利用 WXML 代码在小窗口页面上生成可以与用户交互的图像的过程。

在 WXML 中，我们用 wx:if = "{{condition}}" 控制属性来判断是否需要渲染一个代码块，wx:if 绑定的值为真（true）时开始渲染，例如：

```
<!-- index. wxml -->
<view wx:if = "{{condition}}" >my name is ucle Wong </view >
```

在 wx:if 语句中可以用 wx:elif 和 wx:else 来添加 else 块，wx:else 只能跟在 wx:if 的后面，表示条件判断的一个分支，如果表达式为真则执行对应的渲染语句，例如：

```
<!-- index. wxml -->
<view wx:if = "{{length >7}}" >1 </view >
<view wx:elif = "{{length >4}}" >4 </view >
<view wx:else >32 </view >
```

wx:if 是一个控制属性，使用时需要将它添加到一个组件标签上，如果我们想通过一条判断语句决定几个组件标签是否需要渲染，就可以使用 <block/> 标签将多个组件包装起来，并且使用 wx:if 属性来控制渲染，例如：

```
<!-- index. wxml -->
<block wx:if = "{{true}}" >
<view > view1 </view >
<view > view2 </view >
</block >
```

<block/> 标签并不是一个组件，它仅仅是一个包装元素，它本身不会在页面中做任何渲染，只为控制属性指明对象。

因为 wx:if 之中的模板也可能包含数据绑定，所有当 wx:if 的条件值切换时会有一个局



部渲染的过程，它会确保条件块在切换时销毁或重新渲染。

如果 wx:if 最初的渲染条件为 false，那么小程序什么也不会做，在条件第一次变成真值的时候才开始局部渲染。相比之下，CSS 中 visibility 属性的 hidden 方法就简单得多，组件最初就会被渲染，只是简单的控制显示与隐藏。

所以，wx:if 有更高的切换消耗，而 hidden 方法有更高的初始渲染消耗，如果在页面元素需要频繁切换的场景下，用 hidden 的方式效率会更高，否则用 wx:if 更好。

关于 visibility 属性的 hidden 方法的使用我们会在 WXSS 语法中详细讲解。

8.2　wx:for 列表渲染语句

有时候我们需要对一个组件重复进行渲染，在组件上使用 wx:for 控制属性可以帮我们效率更高地完成这项工作，wx:for 控制属性会绑定一个数组，使用数组中各项数据重复渲染组件。

默认数组的当前项的下标变量名默认为 index，数组当前项的变量名默认为 item：

```
<!-- index. wxml -->
<view wx:for = "{{array}}" >
  {{index}}:{{item. info}}
</view >

//index. js
Page({
  data:{
    array:[{
info:'first - info',
    },{
info:'last - info'
    }]
  }
})
```

在进行组件的列表渲染时，使用 wx:for - item 属性可以指定数组当前元素的变量名，使用 wx:for - index 属性可以指定数组当前下标的变量名：

```
<!-- index. wxml -->
<view wx:for = "{{array}}" wx:for - index = "idx" wx:for - item = "itemName" >
  {{idx}}:{{itemName. ifo}}
</view >
```

wx:for 也可以嵌套，例如下面的代码可以渲染出一个九九乘法表：

```
<!-- index. wxml -->
<view wx:for = "{{[1,2,3,4,5,6,7,8,9]}}" wx:for - item = "i" >
<view wx:for = "{{[1,2,3,4,5,6,7,8,9]}}" wx:for - item = "j" >
<view wx:if = "{{i < =j}}" >
    {{i}} * {{j}} = {{i * j}}
</view >
</view >
</view >
```

类似 <block wx:if/ > 的用法，我们也可以将 wx:for 用在 <block/ > 标签上来渲染一个包含多节点的结构块。例如：

```
<!-- index. wxml -->
<block wx:for = "{{[1,2,3,4,5]}}" >
<view > {{index}} : </view >
<view > {{item}} </view >
</block >
```

如果列表中项目的位置会发生动态改变或者会有新的项目添加到列表中，并且我们希望列表中的项目保持自己的特征和状态（例如 <input/ > 中的输入内容，<switch/ > 的选中状态），就需要使用 wx:key 指定列表中项目的唯一标识符。wx:key 的值以两种形式提供：

- 字符串，代表在 for 循环的 array 中 item 的某个 property（属性），该 property 的值需要是列表中唯一的字符串或数字，且不能动态改变。
- 关键字 this，代表了在 for 循环中的 item 本身，这种表示需要 item 本身是一个唯一的字符串或者数字。

当数据改变触发渲染层重新渲染的时候，会校正带有 key 的组件，框架会确保他们被重新排序，而不是重新创建，以确保使组件保持自身的状态，并且提高列表渲染时的效率。

如果没有设置 wx:key，系统会报一个 warning 信息，如果明确知道该列表是静态，或者不必关注其顺序，可以选择忽略。

示例代码如下：

```
<!-- text. wxml -->
<switch wx:for = "{{obj}}" wx:key = "special" style = "display:block;" > {{item. id}} </
switch >
<button bindtap = "switch" > 交换 </button >
<button bindtap = "addToFront" > 添加到它前面 </button >

<switch wx:for = "{{nbArray}}" wx:key = " * this" style = "display:block;" > {{item}} </
switch >
<button bindtap = "add" > 添加到它前面 </button >
```

```
//text.js
Page({
  data:{
    obj:[
      {id:"火鸡",special:'special_5'},
      {id:"烧鸡",special:'special_4'},
      {id:"野鸡",special:'special_3'},
      {id:"公鸡",special:'special_2'},
      {id:"母鸡",special:'special_1'},
      {id:"小鸡",special:'special_0'},
    ],
    nbArray:["东施","西施","貂蝉","妲己"]
  },
  switch:function(e){
    const length = this.data.obj.length
    for(let i =0;i < length; ++i){
      const x = Math.floor(Math.random() * length)
      const y = Math.floor(Math.random() * length)
      const temp = this.data.obj[x]
      this.data.obj[x] = this.data.obj[y]
      this.data.obj[y] = temp
    }
    this.setData({
      obj:this.data.obj
    })
  },
  addToFront:function(e){
    const length = this.data.obj.length
    this.data.obj = [{id:length,special:'special_' + length}].concat(this.data.obj)
    this.setData({
      obj:this.data.obj
    })
  },
  add:function(e){
    this.data.nbArray = [ this.data.nbArray.length + 1 ].concat(this.data.nbArray)
    this.setData({
      nbArray:this.data.nbArray
    })
  }
})
})
```



课后作业

1. 将我们在本节中讲到的知识点以及列举的案例在微信小程序的开发环境中实现一遍，并观察其效果。

2. 在本节中，我们已经介绍了一个使用微信小程序的方法实现九九乘法表的例子，接下来，请你用同样的方法，实现一个倒三角图形。

第9天
WXML的高级特性

9.1 模板

WXML使用模板（template）来定义代码片段，在WXML文档其他的地方可以调用模板片段。

定义模板的名字使用name属性，然后在<template/>标签块内定义模板的代码片段，也就是模板的内容，例如：

```
<!--index.wxml-->
<!--
  index:int
  msg:string
  time:string
-->
<template name="msgItem">
<view>
<text>{{index}}:{{msg}}</text>
<text>Time:{{time}}</text>
</view>
</template>
```

我们使用is属性声明需要使用的模板，然后将模板所需要的data数据传入，is属性的使用方法如下例：

```
<!--index.wxml-->
<template is="msgItem" data="{{...item}}"/>

//index.js
Page({
  data:{
    item:{
```

```
        index:0,
        msg:'this is a template',
        time:'2016-09-15'
      }
    }
})
```

类似数据绑定，is 属性使用 Mustache 语法来动态决定需要渲染哪个模板：

```
<!-- index.wxml -->
<template name="odd">
<view>odd</view>
</template>
<template name="even">
<view>even</view>
</template>

<block wx:for="{{[1,2,3,4,5]}}">
<template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
</block>
```

模板只能使用 data 传入的数据，我们可以理解为这是模板的作用域。

9.2　事件

在小程序框架里，视图层到逻辑层的通信方式叫作事件（event），事件可以将用户的行为反馈到逻辑层进行处理，事件可以绑定在组件上，当事件被触发，逻辑层中对应的事件处理函数就会被调用。事件对象可以携带额外信息，如 id、dataset、touches 等。

我们通过在组件中绑定一个事件处理函数来使用事件处理，如 bindtap，当用户单击该组件的时候会在该页面对应的 Page 中找到相应的事件处理函数：

```
<!-- index.wxml -->
<view id="tapTest" data-hi="WeChat" bindtap="tapName">Click me!</view>
```

在相应的 Page 定义中写上相应的事件处理函数，参数是 event：

```
//index.js
Page({
  tapName:function(event){
    console.log(event)
  }
})
```

我们可以看到在调试控制台 log 出来的日志信息大致如下：

```
}
"type":"tap",
"timeStamp":895,
"target":{
  "id":"tapTest",
  "dataset":  {
    "hi":"WeChat"
  }
},
"currentTarget":  {
  "id":"tapTest",
  "dataset":{
    "hi":"WeChat"
  }
},
"detail":{
  "x":53,
  "y":14
},
"touches":[{
  "identifier":0,
  "pageX":53,
  "pageY":14,
  "clientX":53,
  "clientY":14
}],
"changedTouches":[{
  "identifier":0,
  "pageX":53,
  "pageY":14,
  "clientX":53,
  "clientY":14
}]
}
```

小程序事件分为冒泡事件和非冒泡事件，这里首先要了解组件的“嵌套”概念，组件的嵌套是指一个组件内部包括了一个或者几个其他的组件，外部组件叫作内部组件的父节点，内部组件叫作外部组件的父节点。

当一个有关组件的事件被触发后，该事件会向父节点传递，这类事件叫作冒泡事件；有关组件的事件被触发后，该事件不会向上传递，这类事件就叫作非冒泡事件。

WXML 的冒泡事件如表 9-1 所示。

表 9-1　WXML 的冒泡事件

类　型	触 发 条 件
touchstart	手指触摸动作开始
touchmove	手指触摸后移动
touchcancel	手指触摸动作被打断，如来电提醒、弹窗
touchend	手指触摸动作结束
tap	手指触摸后马上离开
longtap	手指触摸后，超过 350ms 再离开

除表中列出事件以外的其他组件自定义事件，如果没有特殊申明都是非冒泡事件，如 < form/ > 的 submit 事件、< input/ > 的 input 事件、< scroll - view/ > 的 scroll 事件等。

事件绑定的写法用 key、value 对的形式，与组件的属性写法相同：

- key 以 bind 或 catch 开头，然后跟上事件的类型，如 bindtap、catchtouchstart；
- value 是一个字符串，需要在对应的 Page 中定义同名的函数。否则，当触发事件的时候会报错。

bind 事件绑定不会阻止冒泡事件向上冒泡，catch 事件绑定可以阻止冒泡事件向上冒泡。

例如，在下边这个例子中，单击 inner view 会先后触发 handleTap3 和 handleTap2（因为 tap 事件会冒泡到 middle view，而 middle view 阻止了 tap 事件冒泡，不再向父节点传递），单击 middle view 会触发 handleTap2，单击 outter view 会触发 handleTap1。

```
<!-- index. wxml -->
< view id = " outter"  bindtap = " handleTap1" >
  outer view
< view id = " middle"  catchtap = " handleTap2" >
    middle view
< view id = " inner"  bindtap = " handleTap3" >
      inner view
</view >
</view >
</view >
```

9.3　事件对象

当组件触发事件时，逻辑层绑定这个事件的处理函数会收到一个事件对象用于逻辑层的处理。

BaseEvent 基础事件对象属性如表 9-2 所示。



表 9-2　BaseEvent 基础事件对象属性

属　性	类　型	说　明
type	String	事件类型
timeStamp	Integer	事件生成时的时间戳
target	Object	触发事件的组件的一些属性值集合
currentTarget	Object	当前组件的一些属性值集合

CustomEvent 自定义事件对象属性（继承自 BaseEvent）如表 9-3 所示。

表 9-3　CustomEvent 自定义事件对象属性

属　性	类　型	说　明
detail	Object	额外的信息

TouchEvent 触摸事件对象属性（继承自 BaseEvent）如表 9-4 所示。

表 9-4　TouchEvent 触摸事件对象属性

属　性	类　型	说　明
touches	Array	触摸事件，当前停留在屏幕中的触摸点信息的数组
changedTouches	Array	触摸事件，当前变化的触摸点信息的数组

< canvas/ > 标签中的触摸事件不可冒泡，所以没有 canvas 事件，没有 currentTarget。

type 属性代表事件的类型，timeStamp 属性用于描述页面打开到触发事件所经过的毫秒数，target 属性是触发事件的源组件，如表 9-5 所示。

表 9-5　target 属性

属　性	类　型	说　明
id	String	事件源组件的 id
tagName	String	当前组件的类型
dataset	Object	事件源组件上由 data - 开头的自定义属性组成的集合

currentTarget 是事件绑定的当前组件，包含如表 9-6 所示的属性。

表 9-6　currentTarget 属性

属　性	类　型	说　明
id	String	当前组件的 id
tagName	String	当前组件的类型
dataset	Object	当前组件上由 data - 开头的自定义属性组成的集合

target 和 currentTarget 可以参考上例，当用户单击 inner view 时，handleTap3 收到的事件对象 target 和 currentTarget 都是 inner，而 handleTap2 收到的事件对象 target 就是 inner，currentTarget 就是 middle。

我们可以使用 dataset 的方式在 WXML 文件的组件属性中定义数据，这些数据会通过事件处理传递给逻辑层。dataset 的书写方式是以 data 开头的，多个单词由连字符（ - ）连接，

如果有大写字母，系统会自动转成小写，如 data - element - type，最终在 event. target. dataset 中会将连字符转成驼峰 elementType 方式。

下面是示例代码：

```
<!-- index. wxml -->
<view data - alpha - beta = "1" data - alphaBeta = "2" bindtap = "bindViewTap" > DataSet Test </
view >

//index. js
Page( {
  bindViewTap:function( event) {
    event. target. dataset. alphaBeta === 1      // - 会转为驼峰写法
    event. target. dataset. alphabeta === 2      //大写会转为小写
  }
} )
```

触摸事件的处理用 touches 方法实现，touches 是一个数组，数组中的每一个元素都是一个 Touch 对象（canvas 触摸事件中携带的 touches 是 CanvasTouch 数组）。表示当前停留在屏幕上的触摸点，属性如表 9-7 所示。

表 9-7　当前停留在屏幕上的触摸点的属性

属　性	类　型	说　明
identifier	Number	触摸点的标识符
pageX，pageY	Number	距离文档左上角的距离，文档的左上角为原点，横向为 X 轴，纵向为 Y 轴
clientX，clientY	Number	距离页面可显示区域（屏幕除去导航条）左上角距离，横向为 X 轴，纵向为 Y 轴

CanvasTouch 对象的属性如表 9-8 所示。

表 9-8　CanvasTouch 对象的属性

属　性	类　型	说　明
identifier	Number	触摸点的标识符
x，y	Number	距离 Canvas 左上角的距离，Canvas 的左上角为原点，横向为 X 轴，纵向为 Y 轴

此外，我们还可以通过 changedTouches 的方式定义有变化的触摸点，数据格式与 touches 数组相同。例如，从无变有（touchstart）、位置变化（touchmove）、从有变无（touchend、touchcancel）。

如表单组件提交事件携带的用户的输入数据、媒体的错误事件携带的错误信息数据等自定义事件携带的数据，可以参考本书附录一中组件中各个事件的定义。



9.4　引用

当我们需要在一个 WXML 文件中使用另一个 WXML 文件定义的模板时，就需要使用 WXML 语言的引用语句，WXML 语言提供 import 和 include 两种方式引用其他文件的内容。

import 方式可以在该文件中使用目标文件定义的 template（模板），如事先在 item. wxml 文件中定义了一个叫 item 的 template：

```
<!-- item. wxml -->
<template name = "item" >
<text > {{text}} </text >
</template >
```

在 index. wxml 文件中声明引用 item. wxml 文件，就可以使用这个 item 模板：

```
<!-- item. wxml -->
<import src = "item. wxml"/ >
<template is = "item"  data = "{{text:'forbar'}}"/ >
```

import 有作用域的概念，就是只能引用目标文件中定义的 template，而不能引用目标文件引用的 template，即不能多重引用。

例如，C import B，B import A，在 C 中可以使用 B 定义的 template，在 B 中可以使用 A 定义的 template，但是 C 中不能使用 A 定义的 template：

```
<!-- A. wxml -->
<template name = "A" >
<text > 这是模板 A </text >
</template >

<!-- B. wxml -->
<import src = "a. wxml"/ >
<template name = "B" >
<text > 这是模板 B </text >
</template >

<!-- C. wxml -->
<import src = "b. wxml"/ >
<template is = "A"/ > <!-- 错误,模板 A 尚未引用,不能使用 -->
<template is = "B"/ >
```

include 语句可以将目标文件除了 <template/ > 外的全部代码引入到当前文件，相当于复制了一份目标文件到 include 位置，例如：

```
<!-- index. wxml -->
<include src = "header. wxml"/ >
<view > body </view >
<include src = "footer. wxml"/ >

<!-- header. wxml -->
<view > header </view >

<!-- footer. wxml -->
<view > footer </view >
```



课后作业

1. 使用 template 模板制作中国各省份其下的城市的模板，并在另一个 wxml 文件中引用。

2. 新建一个小程序页面，在 wxml 文件里布置一个 view 标签，在标签里写一个名为 click 的 bindtap 事件，在页面对应的 js 文件里，实现单击 view 区域，给其添加一行文字的效果。

第5部分
WXSS样式表语言

WXSS的全称是WeiXin Style Sheets，意即微信样式表，是一种基于CSS语法规范的样式表语言，用于描述WXML文档中组件的样式，如颜色、边距、高度、宽度、背景图像、高级定位等方面的属性。WXML可以用于为页面添加布局效果，但有可能被误用。而WXSS则提供了更多选择，而且更为精确、高质量地完成渲染任务。可以这么讲，WXML用于结构化内容，WXSS用于格式化结构化的内容。

WXSS与CSS语法的兼容度很高，所以本章内容学完后，也可以把收获应用于其他前端开发的领域里面去，如H5浏览器页面的开发。



本章要点

- WXSS（CSS）基本语法
- 选择器
- 样式表的使用
- CSS属性
- 盒模型
- Flex布局

第 10 天

WXSS 语言基础

10.1 WXSS（CSS）简介

WXSS 基于 CSS 语言架构，几乎所有的 CSS 特性，WXSS 都支持，所以我们从 CSS 讲起。

CSS 是级联样式表（Cascading Style Sheets）的缩写，是一种为页面添加布局效果的出色工具。利用 CSS 可以省去开发者大量定义标签形式的时间，令你可以采用一种全新的高效方式来设计网站。现在，CSS 已经成为每个前端开发者必须掌握的技术。

在互联网刚刚被发明的那个年代，用于给文本添加结构的语言只有 HTML。程序员通过声明标签类型的方式给网页添加文本，如用 p 标签来告诉浏览器“这是段落”。

后来随着 Web 技术的流行，网页设计者们开始产生了为页面增加更多类型的布局的需求，为此，浏览器厂商们开始扩充 HTML 的标签类型，发明了各种新的 HTML 标签，如定义字体的 font 标签，这导致了在 HTML 语言里原本用于结构化的标签被大量地应用于布局，而真正用于布局的标签却得不到广泛应用，这是因为这些用于布局的标签发明的比较晚，大部分浏览器都不支持。所以我们在那个年代访问的网页经常会弹出“您的浏览器不能兼容网页特性”的提示信息。

在这样的环境下，CSS 技术应运而生了，CSS 把页面的布局能力从 HTML 语言中分离了出来，它为页面开发者提供了浏览器能够广泛支持的布局能力。CSS 技术让文档的内容和表现样式分离了开来，也提高了网站的可维护度。

CSS 是一种从结构描述文档中分离出来的专门用来描述结构表现方式的编程语言，它可以用来描述一个组件在屏幕上的布局细节和显示方式。微信小程序采用了内容与样式分离的框架，基于 XML 的 WXML 文档负责内容的组织，基于 CSS 的 WXSS 文档负责显示样式的描述。

CSS 可以设置页面元素的可视属性样式，如颜色、大小、边距等，CSS 的内容与样式分离机制给前端设计带来了很多好处：

- 所有的主流浏览器都支持 CSS；
- 一个样式可以应用在多个元素甚至多个页面（利用外部样式表）；
- 提供了更加精确的布局控制；



- 简单地修改样式表就可以影响所有的相关 HTML 元素，使网站的更新更加容易、迅速。

基于此，我们认为，CSS 的出现是前端设计界的一次革命。从此，内容与样式分离的前端设计思想被确定了下来。

10.2　CSS 基本语法

正如我们 10.1 节所学的，除内联样式外，CSS 语句都是由两部分组成的：选择器和属性声明，如图 10-1 所示。



图 10-1　CSS 的语法结构

（1）选择器（selector）决定了这个样式渲染哪些标签或者组件元素。

（2）被花括号括起来的部分叫作属性声明，属性声明部分是 CSS 语句的主体，决定了被选择器选择的标签或者组件元素采用怎样的方式来渲染，如渲染风格、字体大小、颜色值、空间布局等。

一组属性声明分为两部分：属性名（key）和属性值（value）。属性名和属性值之间用冒号（:）间隔，每一组属性声明都要以分号（;）结束。

声明的语句块要以花括号（{}）括起来，表示这一组属性声明都对应前面的选择器，例如：

```
view{color:#aaa;text-align:center;}
```

我们为了让 CSS 可读性更强，通常每行只描述一个属性：

```
view{
  color:#aaa;
  text-align:center;
}
```

像大部分编程语言一样，CSS 也有仅用于注释的代码机制，注释语句在执行时会被认为是无效语句。

CSS 的注释语句类似 JavaScript，以“/*”开始，以“*/”结束，示例如下：

```
/*这是一行注释语句*/
view{
```

```
    text - align:center;
    /* 这是第二行注释语句 */
    color:black;
    font - family:arial;
}
```

10.3　选择器

微信小程序目前支持的选择器如表 10-1 所示。

表 10-1　微信小程序目前支持的选择器

选　择　器	样　　例	样 例 描 述
. class	. intro	选择所有拥有 class = " intro" 的组件
#id	#firstname	选择拥有 id = " firstname" 的组件
element	view	选择所有 view 组件
element，element	view，checkbox	选择所有文档的 view 组件和所有的 checkbox 组件
:: after	view :: after	在 view 组件后边插入内容
:: before	view :: before	在 view 组件前边插入内容

10.3.1　id 选择器

id 选择器指定的 CSS 样式会对标有指定 id 属性值的 WXML 组件或标签起样式渲染作用，id 选择器的语法规则如下：

```
#id 值{
... 属性声明语句块...
}
```

以“井”号（#）开头的选择器都是 ID 选择器。

例如，下面的样式表规则会对属性 id = “p1” 的组件或标签起作用：

```
#p1{
color:#333;
text - align:left;
}
```

需要注意的是，有些 Web 浏览器支持 id 属性以数字开头，但微信小程序不支持这样的规则。



10.3.2　class 选择器

class 选择器也就是“类”选择器，如果要对一组组件或者标签使用样式表，就要使用 class 选择器。

应用 class 选择器的组件在 WXML 文件中以 class 属性表示。在 CSS 中，class 选择器的语法规则如下：

```
.类名{
... 属性声明语句块...
}
```

与 id 选择器不同，CSS 定义 class 选择器是以圆点符号（.）开头的，后面是 class 名。

如以下的例子，所有包含 cen 类的 WXML 元素文本都将居中显示：

```
.cen{
  text-align:center;
}
```

我们也可以指定某些特定的 WXML 元素使用 class 选择器。

下面的例子中，所有的 view 元素使用 class = “col” 让元素的颜色统一起来：

```
view.col{
  color:#aaa;
}
```

与 id 选择器类似，在小程序中，class 选择器类名的第一个字符不能使用数字。

10.4　使用 CSS 样式表

在 Web 页面里，有三种方法可以让 HTML 文档引用 CSS 样式表来控制标签的显示方式：

- 内部样式定义方式，样式表在 HTML 文档头来定义，也就是在 < head > 标签内定义，采用这种方式，一份 CSS 样式只能控制一份 HTML 文档；
- 内联样式，样式表定义到具体的 HTML 元素，一份 CSS 样式表只能控制一个 HTML 元素；
- 外部样式表，采用 < link/ > 标签或者在 < style/ > 标签中采用@ import 方法来导入样式表文件，这种方法将样式表彻底与 HTML 文件分离，Web 文档需要引用样式要额外导入，一批样式可以控制多个页面的显示，做到了内容与样式的分离。

微信小程序的前端页面内容是由微信自定标准的 WXML 语言描述的，WXML 与 HTML 语言类似，但是没有文件头（head）的语法定义，所以我们只能用内联样式和外部样式表的方法来使用样式表文件。

10.4.1　内联样式

内联 CSS 样式只对单个组件的渲染有效，内联样式常用于精确地控制某个 WXML 视觉组件的样式定义，以及从逻辑层接收动态的样式改变。

小程序的组件支持使用 style 方法使用 CSS 内联样式，定义内联样式的语法格式如下：

```
<组件名 style = "属性 1 :值 1 ;属性 2 :值 2 ;..." >
```

下面是 style 方法控制组件动态样式的一个示例，组件的属性值取自逻辑层的绑定数据，使用了 Mustache 语法，关于数据绑定的细节请查看本书 WXML 语法里数据绑定一节。

```
<!-- index. wxml -->
<view style = "color:{{color}};" />
```

每次页面被调用时都会执行内联样式表，出于代码执行效率的考虑，建议大家在编写静态样式表时尽量利用外部样式表，写到 wxss 文件的 class 选择器中，style 方法只接收动态的样式，以免影响页面的渲染速度。

我们也可以在 wxml 文件内用 class 属性指定样式规则，其属性值是样式表中样式类名的集合，这时样式类名不需要带上圆点符号（.），样式类名之间用空格来分隔：

```
<!-- index. wxml -->
<view class = "normal_view" />
```

10.4.2　外部样式表

小程序提供了样式表独立于 WXML 内容文档的框架，CSS 文档以 . wxss 文件名的形式存在于小程序根目录和页面目录下。

定义在根目录 app. wxss 文档中的样式会作用于小程序的全部页面，叫作全局样式。

定义在 page 页面目录下的 . wxss 文档只会作用于该页面本身，称作局部样式。局部样式被页面使用时会覆盖掉全局样式中相同选择器的内容，也就是说在页面显示时，局部样式的优先级更高。

如在下面的例子中，app. wxss 和页面 . wxss 文件都定义了 page - body 选择器的 width 属性，由于页面 . wxss 的优先级更高，所以 page 页面在显示时会采用 width 为 80% 的属性值：

```
/ ** app. wxss ** /
. page - body{
  width:100% ;
  display:flex;
}
```

```
/ ** page. wxss ** /
. page - body{
    width:80% ;
    display:flex;
}
```

app. wxss 文档和页面 . wxss 文档都定义了 page - body 选择器的 width（页面宽度）属性，由于页面 . wxss 的优先级更高，所以 page 页面在显示时会采用 width 为 80% 的属性值。

“选择器”也就是样式的作用范围，决定样式表会“选择”哪些组件去渲染，我们会在下一节详细介绍。

小程序还支持用@ import 语句导入外联样式表文件，@ import 后跟需要导入的外联样式表的相对路径，用分号（;）表示语句结束，如我们要把“/common”路径下的“common. wxss”样式表文件导入到“app. wxss”样式表文件中来，就可以采用下面的方法：

```
/ ** common. wxss ** /
. p1{
    padding:10px;
}

/ ** app. wxss ** /
@ import "/common/common. wxss" ;
. p2{
    padding:5px;
}
```

10.5　WXSS 的新增特性

微信在定义 CSS 语言规范时，继承了 CSS 语言几乎所有的特性，但由于微信小程序页面主要用于在手机和移动设备的屏幕上展示，WXSS 扩展了 CSS 在显示尺寸单位上的定义。

微信在 CSS 像素的基础上，增加了 rps 的屏幕尺寸定义。

在了解什么是 rpx 之前我们先来了解物理像素和 CSS 像素（css pixel）的概念。

物理像素（device pixel）也就是设备屏幕的实际分辨率，也叫设备像素或者绝对像素，物理像素用 px 单位来表示一个像素点，如 iPhone5 的屏幕物理像素是 640 × 1136px，而目前主流安卓手机的屏幕物理像素是 1080 × 1920px。

如今主流手机屏幕的像素点数已经相等甚至超过桌面显示器的像素数量了，5 英寸 1920 × 1080P 的手机与一个 24 英寸 1920 × 1080P 的显示器相比，同样的单位面积上，手机的像素点要密集好几倍，可想而知手机屏幕上像素点的面积比桌面显示器要小多少。这会给设计页面时造成很大的问题，如在 CSS 中设置一个宽度是 10 物理像素的文字，在手机屏幕和桌面显示器上的尺寸大小就会有很大的差别，桌面显示器能看清的字，到了手机屏幕上可

能就需要用放大镜来看了。

为了让视觉元素在小设备上适当放大，就要用多个物理像素来显示一个逻辑像素，所以 CSS 引入了设备像素比（devicePixelRatio）和“逻辑分辨率“的概念：

逻辑分辨率 = 物理分辨率/设备像素比。

人们定义了常用设备的设备像素比，如普通密度桌面显示屏的设备像素比是 1；高密度桌面显示屏，如 Mac Retina 的设备像素比是 2；主流手机显示屏的设备像素比是 2 或 3。

CSS 像素（css pixel）和我们下面要学习到 rpx、rem 尺寸单位都属于逻辑分辨率的概念范围，指的都是 CSS 样式代码中使用的逻辑像素，但不同的逻辑分辨率单位计算方法有所不同。

例如，iPhone5 使用的是 Retina 视网膜屏幕，设备像素比是 2，也就是说使用 1px 的 CSS 像素代表 2px 物理像素，所以设备像素数为 640 × 1136px，而它的 CSS 逻辑像素数为 320 × 568px。

微信对 CSS 像素又进行了概念上的扩展，为了让小程序更加适应屏幕尺寸和分辨率的碎片化，增加了 rpx 像素作为屏幕尺寸的一个定义标准。

rpx（responsive pixel）叫作“响应式像素”，就是能够根据屏幕宽度进行自适应调整的像素单位。

微信规定所有设备屏幕宽度都是 750rpx。例如，在 iPhone6 上，屏幕的 CSS 像素宽度为 375px，共有 750 个物理像素，那么 750rpx = 375px(CSS) = 750 物理像素，1rpx = 0.5px = 1 物理像素。表 10-2 是一些常用屏幕宽度的 rpx 和 CSS 像素的换算。

表 10-2　一些常用屏幕宽度的 rpx 和 CSS 像素的换算

设　备	rpx 换算 px（CSS px 宽度/750）	px 换算 rpx（750/CSS px 宽度）
iPhone5	1rpx = 0.42px	1px = 2.34rpx
iPhone6	1rpx = 0.5px	1px = 2rpx
iPhone6 Plus	1rpx = 0.552px	1px = 1.81rpx
小米 5	1rpx = 0.72px	1px = 1.39rpx

根据 750rpx 的屏幕总宽度定义来看，微信的 rpx 是参考 iPhone6 屏幕的物理像素宽度定义的，开发微信小程序时设计师可以用 iPhone6 作为视觉稿的参考标准，单位换算起来比较简单、直观。

需要注意的是，在 CSS 文档里，我们用 px 和 rpx 单位描述尺寸单位时，数字和单位之间不要留空格，否则执行时会产生错误。

此外，微信也支持 CSS3 标准新引入的 rem 尺寸单位，微信规定所有的屏幕宽度都是 20rem，所以 rem 与 rpx 的换算关系也就是 1rem = (750/20)rpx。



课后作业

1. 了解一下在 Web 开发中响应式设计的概念，以及实现一个响应式要用到的知识点。
2. 自己动手实现一个简单的响应式 Web 页面。

第 11 天

CSS 常用属性

11.1 尺寸属性

CSS 尺寸（Dimension）属性用来控制元素的高度和宽度，也可以用它来增加行间距。可以设置的属性项如表 11-1 所示。

表 11-1 CSS 尺寸可以设置的属性

属　性	描　述
height	设置元素高度
width	设置元素宽度
line - height	设置行高
max - height	设置元素最大高度
max - width	设置元素最大宽度
min - height	设置元素最小高度
min - width	设置元素最小宽度

设置方法如下例：

```
#p1{
  height:50rpx;
  width:50rpx;
}
```

CSS 尺寸属性可以设置的属性值如表 11-2 所示。

表 11-2 CSS 尺寸属性可以设置的属性值

值	描　述
auto	默认值，窗口自动计算数值
length	使用 rpx、px、rem 等单位定义数值
x%	百分数，定义基于父元素的百分比数值

11.2　背景属性

CSS 背景属性用于定义 WXML 组件的背景呈现。

11.2.1　背景颜色

background - color 属性定义了组件或标签元素的背景颜色，例如：

```
#p1{
  background - color:#2391ad;
}
```

WXML 页面的颜色通常有下面三种定义方式：

- 十六进制的方式，例如，“#13ab88”；
- RGB 颜色值的方式，例如，“rgb（255，0，0）”；
- 颜色名称方式，例如，“red”。

用十六进制表现颜色时，如果第 1、2 位数值相同，3、4 位数值相同，5、6 位数值也相同，可以采用简写的形式，如“#226699”可以简写为“#269”。

WXML 页面的背景颜色在 page 选择器中定义，例如：

```
page{
  background - color:#fbf9fe;
}
```

11.2.2　背景图像

background - image 属性用于描述元素的背景图像。

如果不设置背景图像的覆盖属性，背景图像将平铺重复显示，以覆盖整个元素。下面是设置示例：

```
view{
  background - image:('/page/common/paper. jpg');
}
```

11.3　文本格式

11.3.1　文本颜色

color 属性被用来设置文字的颜色。

```
text{
    color:#00ffaa;
}
```

页面的整体文字颜色用 page 选择器定义，例如：

```
page{
    color:red;
}
```

11.3.2　文本对齐方式

text - align：文本排列属性，用来设置文本的水平对齐方式。
文本可居中（center）或对齐到左（left）或右（right），两端对齐（justify）：

```
#p1{
    text - align:center;
}
#p2{
    text - align:left;
}
#p3{
    text - align:justify;
}
```

11.3.3　文本转换

text - transform 文本转换属性可以用来定义文本中的字符都变成大写（uppercase）或小写字母（lowercase），也可以定义每个单词只有首字母大写（capitalize）：

```
text. uppercase{
    text - transform:uppercase;
}
text. lowercase{
    text - transform:lowercase;
}
text. capitalize{
    text - transform:capitalize;
}
```

11.3.4　文本缩进

text - indent 属性用于定义文本第一行的缩进像素：

```
view{
  text-indent:30rpx;
}
```

11.4　字体属性

CSS 字体属性定义字体的显示效果，如文字样式、字体大小，是否加粗等。

11.4.1　字体系列

font-family 属性设置文本的字体系列，我们通常应该设置几个字体名称作为备用，如果浏览器不支持第一种字体，他将尝试下一种字体。例如：

```
.page-body-text-network-type{
  font-size:80rpx;
  font-family:-apple-system-font,Helvetica Neue,Helvetica,sans-serif;
}
```

几个字体系列中间要用逗号分隔。

11.4.2　字体大小

CSS 用 font-size 属性设置文本的大小，我们可以通过设置字体的 CSS 像素或者微信新增的 rpx 尺寸单位来设置字体的大小，例如：

```
#p1{
font-size:40px;
}
#p2{
font-size:30rpx;
}
```

大家在编写代码时要注意，字体大小的数值与单位（px 或 rpx）之间不能有空格，否则系统会报错。



课后作业

以上介绍到的 CSS 属性，在你的开发环境中实现一遍，并观察其效果。

第 12 天

CSS 布局属性

12.1 盒模型（Box Model）

在了解 CSS 布局能力之前，我们先要了解 CSS 的盒模型，CSS 的布局能力是基于盒模型的。

在 Web 页面里，每个视觉组件可以看作是一个盒子，CSS 提供一个叫作“盒模型”（box model）的概念供我们在设计和布局时使用。

当我们俯视一个盒子时，会看到它有外边距、边框、内边距和实际内容，如图 12-1 所示。

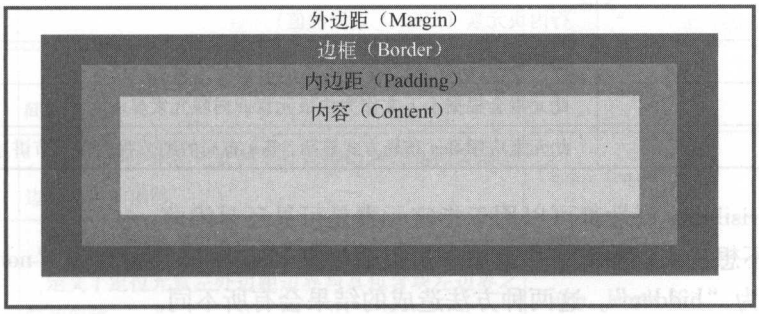


图 12-1　盒模型

CSS 盒模型本质上就是这样一个盒子，它用来封装视觉组件，它包括外边距（Margin）、边框（Border）、内边距（Padding）和内容（Content）四个属性：

- Margin（外边距），边框外的区域，外边距是透明的；
- Border（边框），围绕在内边距和内容外的部分；
- Padding（内边距），也叫填充属性，是指内容周围的区域，内边距也是透明的；
- Content（内容），盒子的实际内容，用于展示页面组件。

盒模型有上下左右四条边，所以每个属性都包括上下左右四个部分，这四部分可以设置一样的参数，也可分别设置；内边距可以理解为盒子里装的东西和边框的距离，而边框有厚薄和颜色之分，内容就是盒子中间装的东西，外边距就是边框外面自动留出的一段空白，内

边距和外边距只有宽度属性。

让我们看看盒模型是如何作用于页面组件布局的：当我们在 CSS 样式表中定义了一个组件的宽度和高度属性时，你只是设置内容区域的宽度和高度。要知道，完全大小的元素，你还必须添加内边距、边框和外边距。

12.2　显示方式与可见性

CSS 用 display 属性设置元素的显示方式，也就是规定元素应该生成的框的类型。

下面的例子使元素对象生成一个行内框：

```
p. inline{
   display:inline;
}
```

display 属性可设置的值及代表的意义如表 12-1 所示。

表 12-1　display 属性可设置的值及代表的意义

值	描　述
none	此元素不会被显示
block	此元素将显示为块级元素，此元素前后会带有换行符
inline	默认。此元素会被显示为内联元素，元素前后没有换行符
inline - block	行内块元素（CSS2.1 新增的值）
list - item	此元素会作为列表显示
run - in	此元素会根据上下文作为块级元素或内联元素显示
flex	此元素应用 flex 布局方式显示，flex 布局的细节我们在下节讲述

display 和 visibility 属性都可以用于指定元素是可见还是隐藏。

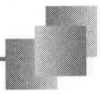
如果我们不想让人们看到一个元素，可以通过把 display 属性设置为“none”，或者 visibility 属性设置为“hidden”。这两种方法造成的结果会有所不同。

（1）display：none 隐藏某个元素的同时会释放元素占用的页面空间。

（2）visibility：hidden 隐藏某个元素时，元素占用的空间不会被隐藏，也就是说元素的布局依然在起作用。

下面是两种方法的示例代码，请大家把代码布置在页面里分别尝试一下两种方法带来的显示效果上的差别：

```
#p1. hidden{
visibility:hidden;
}
#p2. hidden{
display:none;
}
```



12.3　position 定位

CSS 用 position 属性定义元素的定位类型。

position 属性可设置的值及代表的意义如表 12-2 所示。

表 12-2　position 属性可设置的值及代表的意义

值	描　述
absolute	生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过“left”、“top”、“right”及“bottom”属性来定义
fixed	生成绝对定位的元素，相对于设备显示窗口进行定位。元素的位置通过“left”、“top”、“right”及“bottom”属性来定义
relative	生成相对定位的元素，相对于其正常位置进行定位。因此，“left:10rpx”会向元素的 LEFT 位置添加 10rpx 像素
static	默认值。没有定位，忽略样式表中 top、bottom、left、right 等属性值对定位的影响

position 属性被定义后，元素的其他定位属性，如 left、top 等属性才能发挥作用。

CSS 其他用于定位的属性如表 12-3 所示。

表 12-3　CSS 其他用于定位的属性

属　性	描　述	可　选　值
bottom	定义了定位元素下外边距边界与其包含块下边界之间的偏移	auto、length、%
top	定义了一个定位元素的上外边距边界与其包含块上边界之间的偏移	auto、length、%
clip	剪辑一个绝对定位的元素	shape、auto
left	定义了定位元素左外边距边界与其包含块左边界之间的偏移	auto、length、%
right	定义了定位元素右外边距边界与其包含块右边界之间的偏移	auto、length、%
overflow	设置当元素的内容溢出其区域时发生的事情	auto、hidden、scroll、visible
overflow - y	指定如何处理顶部/底部边缘内容溢出元素的内容区域	auto、hidden、scroll、visible、no - display、no - content
overflow - x	指定如何处理右边/左边边缘内容溢出元素的内容区域	auto、hidden、scroll、visible、no - display、no - content

其中 auto、length、% 等尺寸值表述的意义我们在“CSS 尺寸”一节已经做过讲解。

以下分别是利用 fixed 和 relative 属性实现绝对定位和相对定位的代码示例：

```
#p1. pos_fixed{
position:fixed;
top:20rpx;
right:10rpx;
}
#p2. pos_left{
position:relative;
left:-10rpx;
}
```

12.4　float 浮动属性

float 属性用于图像浮动的设置，定义元素在哪个方向浮动。在 CSS 中，这个属性不仅仅用于图片，任何组件都可以设置浮动，浮动元素会生成一个块级框。

浮动属性可选择的值及会带来的效果如表 12-4 所示。

表 12-4　浮动属性可选择的值及会带来的效果

值	描　述
left	元素向左浮动
right	元素向右浮动
none	默认值。元素不浮动，并会显示在其在文本中出现的位置

下面的代码，定义了图片向右浮动：

```
image{
  float:right;
}
```

12.5　边框

我们学习过了盒模型的概念，在 CSS 盒模型里，元素的边框（border）是围绕在元素内容和内边距周围的一条或多条线，border 属性用来设定元素边框的样式、宽度和颜色。

12.5.1　边框样式

边框样式（border-style）属性指定要显示什么样的边界，边框样式属性可设置的值及表现的效果如图 12-2 所示。

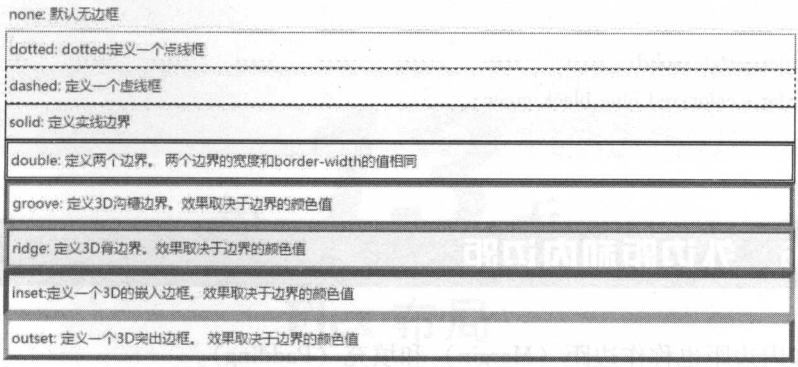


图 12-2　CSS 边框样式属性

12.5.2　边框宽度

CSS 通过 border - width 属性为边框定义宽度。

border - width 属性既可以为边框指定长度像素值，如设置成 1rpx 或者 2px；也可以用长度关键字设置，如设定值为 thick、medium（默认值）和 thin。

CSS 并没有定义三个关键字的具体宽度，所以你可以自主设定这三个值的大小，如把 thick、medium 和 thin 分别设置为 6rpx、2rpx 和 1px。

下面是边框宽度设置的示例：

```
#p1{
  border-style:solid;
  border-width:5rpx;
}
#p2{
  border-style:solid;
  border-width:thick;
}
```

12.5.3　边框颜色

border - color 属性用来设置边框的颜色，我们可以用前面学过的颜色表示方法定义边框的颜色。

border - color 属性一次可以接受最多四个颜色值，分别定义上、左、下、右边框的颜色。

需要注意的是，必须先使用 border - style 设置边框样式，边框颜色属性才能起作用。

例如，下面的代码，给一个实线边框设置了红、蓝、黑、绿四种颜色：

```
#p1{
  border-style:solid;
  border-color:red blue black green;
}
```

12.6　外边距和内边距

外边距和内边距也称作边距（Margin）和填充（Padding）。

CSS 用 margin 属性定义元素周围的空间，也就是外边距；用 padding 属性定义元素内容和元素边框之间的空间，也就是内边距。

margin 和 padding 没有背景颜色，完全透明，它们可以定义的是长度单位。

可以定义的值有 auto、length 长度（单位是 rpx 或 px 等）及百分比（%）。

下面是外边距和内边距定义的两个示例：

```
#p1{
  margin:10rpx;
}

#p2{
  padding:10rpx 5rpx 15rpx 0rpx;
}
```

p1 用一个值定义了四边宽度相等的外边距，p2 用四个值定义了四边宽度不等的内边距，定义的顺序是上、左、下、右。

课后作业

如果在边框属性中我们只设置了三个 padding 值，在浏览器中的效果会是怎么样的？实际操作一下，观察其效果，看看与你的猜想是否一致。

第 13 天

Flex 布局

13.1 Flex 布局简介

前面我们学习的 CSS 布局方案，都是基于盒子模型，依赖 display、position、float 的传统布局解决方案。

但是盒子模型也有它的不足之处，如我们需要元素垂直局中布局的时候或者设计元素根据屏幕形式自动伸缩的响应式页面的时候，仅仅依靠盒子模型布局方式就很难实现了。

为了解决这个问题，Flex 布局方案出现了，它可以简单地实现各种响应式布局，很快就成为了 CSS 布局的首选方案，如微信小程序的 Demo 就是基于 Flex 布局方案实现的。

Flex 是 Flexible Box 的缩写，意即弹性布局，也被称为柔性布局，它能让子项目方便地改变宽度、高度和顺序，为盒子模型提供最大的布局灵活性。

Flex 布局的元素，其大小是可伸缩的，可以根据需要扩展自己的尺寸来填满可用空间，或缩小它们，避免溢出容器。

在常规的盒子模型布局中方向是确定的，块就是从上到下，内联就从左到右，而在 Flex 布局中，方向是不可预知的。当我们在设计页面时涉及元素的缩放、拉伸收缩、方向改变的场合，使用 Flex 布局就能感受到它的灵活性了。

13.2 模型

采用 Flex 布局的元素，称为伸缩容器（flex container），简称“容器”。它的子元素，也就是容器成员，称为伸缩项目（flex item），简称“项目”，Flex 布局的模型如图 13-1 所示。

伸缩容器存在着两根轴：水平的主轴（main axis）和垂直的侧轴（cross axis）。主轴的起始位置叫作 main start，结束位置叫作 main end；侧轴的起始位置叫作 cross start，结束位置叫作 cross end。

伸缩项目沿着容器的主轴排列，项目在主轴上占据的空间就是主轴尺寸（main size），占据的侧轴空间叫作侧轴尺寸（cross size）。

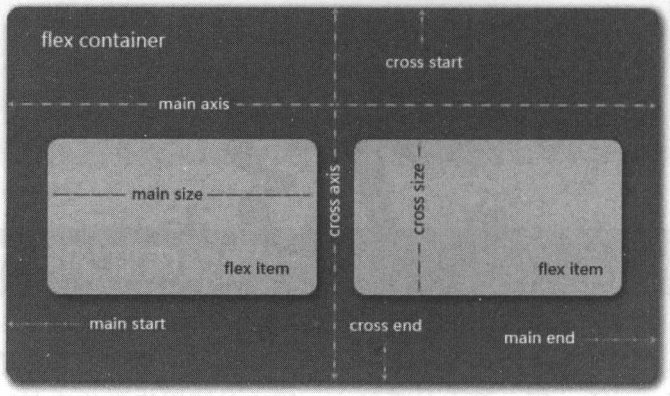


图 13-1　Flex 布局模型

主轴不一定是水平的，这取决于“justify - content”属性的值，我们将会在下文详细介绍。

项目主要的大小属性，也就是主轴尺寸，要么是项目的宽度（width），要么是高度（height），具体是哪个要看哪个值对应着主轴的方向。

页面的任何一个容器都可以指定为 Flex 布局，例如：

```
.mainBox{
  display:flex;
}
```

行内元素也可以使用 Flex 布局，例如：

```
.mainBox{
  display:inline-flex;
}
```

Flex 布局方式的使用是在组件的 display 属性里设置的：

```
display:flex;
```

这条语句用来定义一个元素使用 Flex 布局。

13.3　容器属性

13.3.1　主轴方向属性

flex - direction 属性用来创建主轴，决定主轴的方向，也就是项目的排列方向。

可选的值如图 13-2 所示。

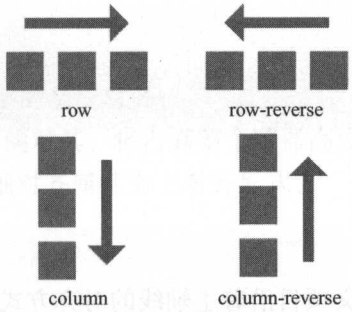


图 13-2　主轴方向

- row：默认值，主轴为水平方向，从左向右；
- row - reverse：主轴从水平方向，从右向左；
- column：主轴为垂直方向，从上到下；
- column - reverse：主轴为垂直方向，从下到上。

例如，下面的代码：

```
.box{
  flex-direction:column;
}
```

定义了主轴的方向是“从上到下”。

13.3.2　换行属性

flex - wrap 属性定义了项目是单行显示还是多行显示，类似 windows 记事本的“自动换行”功能。

可选的值如下所示。

- nowrap：默认值，不换行；
- wrap：换行，由上到下排列，项目的第一行在最上方；
- wrap - reverse：换行，由下到上排列，项目的第一行在最下方。

例如，下面的代码定义了容器由上到下排列的换行方式：

```
.box{
  flex-wrap:wrap;
}
```

13.3.3　简写形式

flex - flow 属性是 flex - direction 和 flex - wrap 属性的简写形式，默认值为 row nowrap。

例如，下面用一行代码定义了从左到右的主轴方向和由上到下的换行方式：

```
.box{
  flex-flow:rowwrap;
}
```

13.3.4　主轴对齐属性

justify-content 属性用来定义项目沿着主轴线的对齐方式。
可选的值如图 13-3 所示（假设主轴方向是从左到右）。

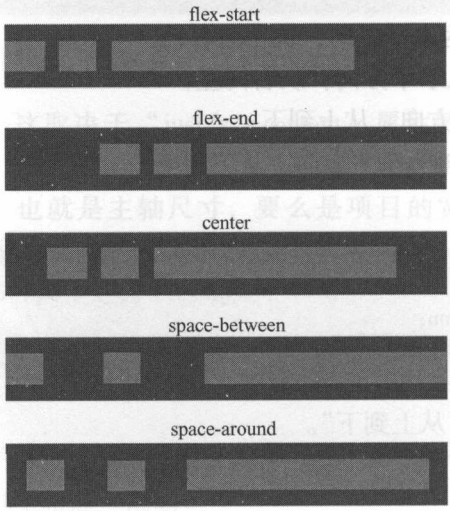


图 13-3　项目对齐方式

- flex-start：默认值，左对齐；
- flex-end：右对齐；
- center：居中对齐；
- space-between：两端对齐，项目间的间隔相等；
- space-around：项目等距分布在行里，两端保留项目间距一半的空间。

例如，下面的代码定义了项目的居中对齐：

```
.box{
  justify-content:center;
}
```

13.3.5　侧轴对齐属性

align-items 属性定义项目在侧轴上的对齐方式。可选的值（假设侧轴的方向是从上到下）如下所示。

- flex-start：上对齐；



- flex – end：下对齐；
- center：上下局中对齐；
- baseline：项目第一行文字的基线对齐；
- stretch：默认值，上下两侧对其，将占满整个容器高度。

例如，下列代码设置了侧轴对齐属性为上对齐形式：

```
.box{
  align-items:flex-start;
}
```

13.3.6　多行对齐属性

align – content 属性定义了多行项目的对齐方式。如果只有一行项目，该属性没有效果。该属性可设定的值有六个，如下所示。

- flex – start：与侧轴的起点对齐；
- flex – end：与侧轴的终点对齐；
- center：与侧轴的中点对齐；
- space – between：与侧轴的两端对齐，行间隔平均分布；
- space – around：各行在侧轴中平均分布，两端保留行间距一半的空间；
- stretch：默认值，各行占满整个侧轴。

13.4　项目的属性

13.4.1　顺序属性

order 属性定义了项目的排列顺序。数值小的排列在前面，order 属性的默认值为 0。下面是一个设置项目 order 属性的示例：

```
.item1{
  order:1;
}
```

13.4.2　放大比例属性

flex – grow 属性用来定义项目的伸缩放大比例，默认值为 0，即不管有没有剩余空间都不放大，如下面是把项目放大两倍的代码：

```
.item{
  flex-grow:2;
}
```

如果一个容器里所有项目的 flex-grow 属性值都为 1，那么每个项目都将等分剩余空间。如果你给其中一个项目的 flex-grow 属性设置为 2 了，其他项目不变，那么这个改变了 flex-grow 属性值的项目占据的剩余空间就会比其他项目多一倍。

13.4.3　缩小比例属性

与 flex-grow 属性的作用相反，flex-shrink 属性定义了项目的缩小比例，默认值为 1，即如果空间不足，项目将缩小：

```
.item{
  flex-shrink:0;
}
```

如果一个容器里所有项目的 flex-shrink 属性值都为 1，那么当空间不足时，每一个项目都将等比例缩小。如果其中一个项目的 flex-shrink 属性为 0 了，其他项目不变，那么空间不足时，前者不缩小，其他项目会缩小。

13.4.4　伸缩基准属性

flex-basis 属性定义了在分配剩余空间前项目占据的主轴空间（main size）。浏览器将根据 flex-basis 的属性值计算主轴的剩余空间，以决定项目是否缩放，怎样缩放。它的默认值是 auto，也就是项目的本来大小。

如果 flex-basis 设置了固定的像素值，如 100rpx，那么项目将占据固定的空间，不理会浏览器对项目的缩放。

下例设置了项目的基准空间为 60rpx：

```
.item{
  flex-basis:60rpx;
}
```

13.4.5　简写形式

flex-grow、flex-shrink 和 flex-basis 三个属性可以统一简写为 flex 属性，默认值为[0, 1, auto]。

在 flex 属性的设置语法里，只有 flex-grow 属性值是必须的，其他两项值可以默认。

例如，下面的代码设置了 flex-grow 的属性值为 2、flex-shrink 的属性值为 1、flex-basis



的属性值为 100rpx：

```
.item{
  flex:2 1 100rpx;
}
```

13.4.6　单项对齐属性

align - self 属性用于设置单个项目对齐方式，设定值可以与 align - items 属性设置的其他项目的对齐方式不同。默认值是 auto，表示继承容器的 align - items 属性。示例如下：

```
.item{
  align - self:flex - start;
}
```

align - self 属性的可选值有 6 个，除了 auto 以外，都与 align - items 属性的可选值一致。



课后作业

在微信小程序开发工具里把上述实现一遍，观察其效果的不同。

第6部分
JavaScript 入门

微信小程序的逻辑层代码基于 JavaScript 语言，本章将会带你走进 JavaScript 的精彩世界，建立程序员的思维方法。

JavaScript 是一种脚本语言，它是为 Web 应用开发而生的，它名为“Java 脚本”，但它与 Java 语言没有什么关系。JavaScript 在客户端浏览器上运行，它是一种面向对象的编程语言，基于对象和事件驱动，动态解释执行。

在 JavaScript 出现之前，Web 网页都是静态内容，就像是一张张的纸，只能阅读上面的内容，不能做任何交互。JavaScript 是为了扩展页面功能而生的，它拥有动画渲染能力和用户交互支持，这使得 Web 页面拥有了越来越多的功能，越来越具有生命力。



本章要点

- 语句
- 变量和数据类型
- 数组
- 条件语句
- 循环语句

第 14 天

JavaScript 语言基础

14.1 文件的编辑和运行

最适合微信小程序编程的编辑器和运行器是微信小程序开发工具（IDE），我们建议初学小程序开发的尽量使用小程序 IDE 作为 JavaScript 的编译器和调试工具。

当然，因为几乎所有网页浏览器都支持 JavaScript 脚本的解释执行，也因为 JavaScript 不需要编译成二进制代码的特性，我们也可以使用 notepad + + 和 Windows 记事本等流行的文本编辑器来编辑 JavaScript 脚本，采用对 JavaScript 支持良好的 Chrome 浏览器来运行脚本，这些编辑器和浏览器都可以很容易地在网上下载安装。

与其他网页浏览器的框架类似，小程序的 JavaScript 文件同样是以 .js 文件的形式存在的，小程序的 JavaScript 页面有两种：定义小程序本身的 app.js 和页面的控制逻辑文件 page.js，它们一起构成了小程序的逻辑层，负责小程序的数据运算、事件处理及与服务器的通信等。

在 Web 页面系统里，JavaScript 代码可以嵌入在 HTML 文件代码里使用，也可以单独保存在 .js 文件里供 HTML 文件调用，如下面的两种方式。

- 第一种方法是采用下面的代码，把 JavaScript 语句直接包含在 HTML 文件里：

```
<! --index.html-->
<script>
//...JavaScript 语句...
</script>
```

- 第二种方法是把 JavaScript 代码保存在单独的 .js 文件里，然后在 HTML 文件里用 <script> 标签的 src（源）属性把这个 .js 文件包含到页面里：

```
//common.js
<script src='/pages/TypeEnglish/myJScode.js'></script>
```

这段代码调用了一个名叫 myJScode.js 的文件，它位于“/pages/TypeEnglish/”路径下面，如果 .js 文件和调用它的 HTML 文件在同一路径下面，可以省略路径的描述。



而微信小程序采用了流行的结构、样式和控制逻辑分离的框架，每一个页面的 JavaScript 逻辑文件都与定义页面框架的 WXML 和定义页面样式的 WXSS 文件并列放置。这样，程序的架构更为清晰，数据的通信也更为简单。

小程序的 JavaScript 与 Web 浏览器兼容的 JavaScript 是有区别的，因为小程序只针对移动端开发，与网页应用的环境有所差别，所以小程序不支持 JavaScript 的 window 和 DOM 对象，如果我们对小程序外的前端设计有兴趣，window 和 DOM 对象都是很重要的，建议大家在网络上找一些辅助材料学习。

14.2　语句

一个 JavaScript 程序是由一些指令代码组成的，这些指令代码叫作 JavaScript 的“语句”，语句是 JavaScript 的基本执行单位。

JavaScript 语言可以用每行放置一条的形式来组织语句，例如：

```
语句1
语句2
语句3
```

也可以在同一行里放置几条语句，所有的语句都以分号（;）结尾，例如下面的形式：

```
语句1;语句2;语句3;
```

但是为了减少输入语句时出错的可能，也为了提高代码的可读性，让程序的调试变得更方便，我们建议采用每行一条语句，以分号（;）结尾的方式来书写 JavaScript 语句：

```
语句1;
语句2;
语句3;
```

14.2.1　语句块

语句块就是使用花括号括起来的多个语句，语句块往往是为了实现某种功能的一组语句的集合，我们可以把它当作一个单独的语句。下面是一个语句块的示例：

```
{
  x = a * b;
  y = a/b;
  console. log("打字学英语");
}
```

语句块中的每个语句都需要以分号结束，语句块的花括号后不需要以分号结束。

14.2.2　注释语句

为了编辑调试代码方便和增加代码的可读性，我们可以在 JavaScript 代码里加入注释语句，浏览器不会执行注释语句的内容。

如果注释语句只有一行，我们可以采用如下的方式来添加注释：

```
//注释内容
```

如果需要添加多行注释，我们也可以采用如下的方式，即使用/*标签来标示注释的开始，使用*/标示注释的结束：

```
/*这行和下面的两行都是注释内容
  这是第二行注释内容
  这是第三行注释内容*/
```

编程时添加注释是个很好的习惯，可以作为给我们自己的提示，在编辑和调试时发挥作用，多人协作时会尤其有用。

14.2.3　小程序的调试输出语句

调试输出语句也叫打印语句，目的就是把信息打印在屏幕上，调试输出语句供我们在调试代码时检查程序执行状态使用，我们需要在调试中了解程序的执行进度或者变量的状态，可以使用 console. log 方法进行，语法如下：

```
console. log(变量名)
```

或者：

```
console. log("字符串")
```

语句被执行后我们就会在小程序的调试控制台窗口看到变量值或者字符串值被打印了出来。



课后作业

1. 写一个语句块，里面放置两条注释语句，注释的内容分别是“JavaScript 的基本执行单位是语句”和“注释语句的实现有两种方式”。

2. 写一段代码，在调试控制台输出一行字符：“学习程序设计要常常练习，坚持不懈”。

第 15 天

变量与数据类型

15.1 变量

几乎所有的编程语言都有变量的概念，变量是数据处理的基础。

我们可以把变量看作一个用来保存数据的容器。

变量中保存的数据可以有多种形式，如是一个字符串“打字学英语 app”或者一个数字256，也可以是其他的形式，变量的形式叫作数据类型。

JavaScript 要求我们使用字母、数字、$符号和下画线给变量命名。在命名方式上，程序员一般使用一种叫作“驼峰命名法”的方法给变量命名，就是把代表变量意义的几个单词或者短语连写在一起，不要有空格，第一个单词的首字母小写，后面单词的首字母大写，如appName。

把数据保存在变量里的过程叫作给变量“赋值”，变量的赋值操作是通过赋值符号（=）进行的，例如：

```
appName = "TypeEnglish";
```

这行语句把“TypeEnglish”这个字符串赋值给了变量 appName。

需要注意的是，数值型的数据可以直接用于变量的赋值，当字符串数据赋值给变量时，必须用双引号或者单引号把字符串括起来。JavaScript 语言里所有对字符串的操作都需要采用这种把字符串用引号括起来的方式。

JavaScript 是一种弱数据类型编程语言，就是说，不需要声明数据类型就可以直接使用变量，但是为了培养良好的编程习惯，我们建议采用 var 关键字申明变量，如上例的语句我们可以写成：

```
var appName = "TypEnglish";
```

变量赋值后，我们就可以用变量把这个值传递出去给程序使用，如可以传递给打印语句 console. log：

```
console. log(appName);
```

语句执行后，我们就能在小程序的调试控制台看到变量值被打印出来。

15.2　操作符

JavaScript 可以使用算术操作符对数值数据进行加、减、乘、除运算，例如：

```
var num = 8 + 9;
var totalNum = 4 * 9;
var singlePrice = totalNum/mum;
```

JavaScript 的加减乘除运算操作符分别是“ + ”、“ - ”、“ * ”、“/”，此外也可以用“%”运算符计算除法的余数，求余数的运算也被称为求“模”运算，如下面的例子：

```
varlastNum = 43 % 20;
console. log( lastNum) ;
```

lastNum 变量最终的计算结果将是数值 3。

变量值的增加和减少也可以采用 JavaScript 语言的自加（++）和自减（--）运算符，例如：

```
deskCount ++ ;
```

相当于：

```
deskCount = deskCount + 1;
```

JavaScript 可以用“ += ”和“ -= ”运算符计算增加或减少的值不为 1 的情况，例如：

```
total = total + 3;
total  += 3;
```

这两条语句的运算效果是一样的。

除数值外，字符串也可以进行相加（+）操作，结果是把运算符两边的字符串连接起来，例如：

```
var myFirstName = " 老" ;
var myLastName = " 王" ;
var myFullName = myFirstName + myLastName;
console. log( myFullName) ;
```

输出的结果将会是字符串“老王”。

我们在一条 JavaScript 的语句里经常会用到多个运算符，哪一个运算符先执行是依据语言的优先级原则进行的，如下面的语句：

```
var a = b + c - d * e/f;
```

在这条语句里，乘法与除法的优先级是同级的，高于另外一组同级的加法与减法，所以会进行 d * e/f 的运算后再被 b、c 相加和相减。

15.3　数据类型

JavaScript 是一种弱类型脚本语言，对数据类型的定义比较宽松，不必事先声明变量的数据类型就可以使用变量。

JavaScript 的基本数据类型有 5 种：数值、字符串、布尔型、null（空）、undefined（未定义）。我们还会在本节学习 JavaScript 强大的数据结构中的一种：数组对象。

15.3.1　数值

数值也就是我们在数学中应用的数据类型，数值包括整数和浮点数。

整数型数值如 0、1、-20 等。

浮点数也就是带有小数部分的数值类型，如 2.1,3.1415,-99.1 等，无限小数也是浮点型数值，如 π 的值 3.1415926……。

浮点数可以表示成传统的形式，也可以表示成指数的形式，例如 21.4e2。

15.3.2　字符串

字符串由一个或多个的一串字符构成，也经常被称作文本类型，字符串必须用一对单引号或双引号括起来，例如：

```
var myStr1 = "打字学英语 app";          //双引号标注方式
var myStr2 = '打字学英语 app';          //单引号标注方式
```

有些字符是无法在键盘上打出来也无法显示在手机或者电脑屏幕上的，如换行符、制表符号等，针对这种字符，编程语言提供了一种叫“转义序列”的组合方式，让程序可以看得懂这些字符，常用的转义字符有如下几种，如表 15-1 所示。

表 15-1　常用的转义字符

转义序列	代表的字符	转义序列	代表的字符
\n	换行符	\\	反斜线
\t	制表符	\0mnn	八进制数 0mnn 代表的字符
\'	单引号	\xhh	十六进制数 0xhh 代表的字符
\"	双引号	\uhhhh	十六进制数 0xhhhh 代表的 Unicode 字符

JavaScript 有一些比较重要的用于处理字符串对象的函数：

- concat( ) 函数用于几个字符串的连接；
- indexOf( ) 函数返回指定值在字符串里出现的第一个位置，用于数据的检索；
- lastIndexOf( ) 函数返回指定值在字符串里出现的最后一个位置；
- replace( ) 函数在一个字符串里搜索指定的子字符串，用新的子字符串去替换；
- split( ) 函数把字符串分解为一系列子串，保存到数组里，返回值是一个新的数组；
- substr( ) 函数用于从指定位置开始，提取指定数量的字符组成新字符串；
- toLowerCase( ) 函数把字符串转换为小写；
- toUpperCase( ) 函数把字符串转换为大写。

15.3.3　布尔型

布尔型数据只有两个值，true 表示（真），false 表示（假），用于保存逻辑操作的结果。例如：

```
var suc = true;
```

布尔型数据用于数值计算时，JavaScript 会把 true 转换为 1，false 转换为 0。

一个布尔型的变量，把“非操作符”（!）放置于它前面，表示逻辑的“非”，也就是“相反”，例如：

```
var suc = true;
console. log( !suc) ;
```

我们会看到在调试控制台时会有 false 值提示。

15.3.4　数据类型转换

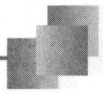
JavaScript 语言支持数据类型的自动转换和利用数据类型转换函数强制转换两种方式。自动转换编程时会很方便，不用额外写代码，但是代码的可读性很差，例如：

```
var a = "6. 2";
var b = a - 3;
var c = a + 3;
```

计算结果是 b 的值是数值 3. 2，c 的值是字符串 6. 23。

在这段代码中，a 的值是个字符串，让 a 与一个数值做减法，因为字符串类型的数据无法简单相减，系统会把 a 转换为数字类型再做运算，所以得到的结果是一个数字值；当让字符串 a 和一个数字做加法时，系统会把数字自动转换成字符串再相加，所以得到了“6. 23”的字符串值。

我们看到，这种让系统自动转化数据类型的方式，可读性非常差，容易得到我们不需要



的意想不到的计算结果，所以我们建议采用 JavaScript 提供的数据类型强制转换方法。

JavaScript 提供了如下函数用于数据类型转换：

- toString( ) 函数用于将布尔值和数值转换成字符串，转换结果是一个 object 对象，关于对象的概念，我们会在后续的章节讲述；
- parseFloat( ) 函数可以把字符串中的数字提取出来，返回一个浮点数；
- parseInt( ) 函数与 parseFloat( ) 函数功能类似，提取数字的整数部分出来，返回整数值，不做四舍五入运算。

例如，parseFloat（“80.6 公斤”）会返回数值 80.6，而 parseInt（“80.6 公斤”）会返回 80。

15.3.5　null 和 undefined

undefined 表示变量未被定义，变量不存在或者对象的属性不存在。它只有一个同名的值：undefined。

unll 表示变量是已经被定义了的，但是没有被赋值，值为空。

一般情况下 null 值与 undefined 值相等，如果要判断变量是否定义可以使用绝对等于符号（===）。

比如下面的语句，变量 a 被赋值了 String 类的一个不存在的属性 aaa，所以变量 a 是未被成功定义的，我们如果引用变量 a 的值，就会得到一个 undefined 数据：

```
vara = String. aaa;
```

15.4　数组

数组是一个打包了的容器，可以理解为数据的组合，数组是 JavaScript 提供的一种复合数据类型。

一个数组变量里保存多个数据，这些数据的类型可以是多种，如数值、字符串、布尔值等，甚至可以是高级数据类型，如其他数组、函数、对象等。

数组里的每一个数据都有一个数值索引，记录了数据在数组中的位置，索引号是从 0 开始的，而不是 1，这一点需要我们注意。

15.4.1　创建和初始化数组

JavaScript 提供了两种创建数组的方法，我们可以用创建对象的方法来创建数组：

```
var myArray = new array( );
```

也可以采用另外一种更方便的方法，用方括号（[ ]）创建数组：

```
var myArray = [ ];
```

创建数组时也可以进行数组的初始化，也就是给数组赋初值，例如：

```
var myArray = ['day','month','year'];、
```

我们在创建这个数组的同时添加了 day、month、year 三个元素。

也可以在创建数组后再赋值，例如：

```
var newArray = [ ];
newArray[0] ='day';
newArray[1] ='month';
newArray[2] ='year';
```

数组有长度属性，也就是数组中元素的个数，用 array. lenth 表示，如上例，数组的长度就是 3。

15.4.2　数组元素的存取

可以使用与赋值时相同的符号读取数组的内容，例如：

```
var newArray = ['day','month','year'];
time = " newArray[0]";
console. log( time);
```

控制台窗口就会列出数组元素的内容。

15.4.3　数组的操作

JavaScript 提供了一系列用于数组操作的函数，调用这些函数的格式是用“数组名．函数名”的方式，中间的点就是操作符，“.”操作符的方式广泛地应用在面向对象的编程语言里，是 JavaScript 操作对象的属性和方法的通用方式。

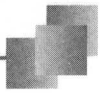
concat( )方法用于合并多个数组，例如：

```
varmyTime = ['hour','minute'];
var myDate = ['day','month','year'];
varmyArr = myDate. concat( myTime);
```

myArr 数组包含的元素将会是 day、month、year、hour、minute。下面的方法我们都会通过这几个被定义的数组来演示用法。

join( )函数可以把数组的元素合并到一个字符串里，join( )函数支持传递一个参数，用于元素之间的分割，例如：

```
varnewTime = myTime. join(" ---");
```



newTime 的值将会是“hour - - - minute”，字符串类型。

如果 join( ) 函数的参数为空，那么结果将会是“hourminute”。

toString( ) 函数与 join( ) 函数类似，同样以字符串的形式返回数组，不同之处是数组元素之间会用逗号分割：

```
varnewTime = myTime. toString( );
```

这时 newTime 的值就是字符串“hour，minute”。

indexOf( ) 函数用于在数组里搜索指定元素，返回搜索到的第一个元素的索引值，如果没有找到，则返回 -1：

```
vara = myTime. indexOf('hour);
varb = myTime. indexOf('second');
```

a 的值将会是 0，b 的值将会是 -1。

lastIndexOf( ) 函数与 indexOf( ) 函数的效果相反，返回搜索结果最后一个元素的索引值。

slice( ) 函数会传递两个参数，根据指定的索引和长度返回一个新的数组，第一个参数是开始的索引值，第二个参数是长度，一般用于提取元素：

```
varnewDate = myDate. slice(1,2);
```

newDate 数组包含的内容将会是 month 和 year。

sort( ) 函数用于排序，可以把数组元素按照字母顺序排列：

```
myArr. sort( )
```

排序的结果将会是 day、hour、minute、month、year。

splice( ) 函数可以在数组里添加或删除指定的元素，它的语法形式是：

```
array. splice(参数 1,参数 2,[参数 3])。
```

第一个参数指定开始的位置，第二个参数说明要删除的元素个数，第三个参数可选，表示要插入的新元素列表，例如：

```
myArr. splice(1,2,"second")
```

表明从索引值为 1（也就是元素列表的第 2 个）的元素开始删除，删除两个元素，插入一个“second”元素，最终数组 myArr 的元素将会是：day、second、hour、minute。



课后作业

1. 布尔类型的值都有哪些？表示“非”逻辑关系的运算符是什么？

2. 一数组的内容是 [“Monday”，“Tuesday”、“Wednesday”、“Thursday”]，使用 indexOf（‘Wednesday’）方法的返回值是什么，是什么数据类型？

第16天 流程控制语句

16.1 条件语句

程序中的语句一般按顺序运行，我们有时候会想让程序做一个判断，然后决定下一步的流程，有时候我们需要程序重复执行固定的几条语句，因此顺序运行的程序远远不能满足我们的需求。

大多数编程语言都提供了流控制语句来控制代码执行的顺序，今天我们将介绍 JavaScript 中最重要的两种流程控制语句：条件语句和循环语句。

条件语句也叫分支语句，它是一种判断条件是否满足，并决定去做什么的语句，JavaScript 的条件判断是依靠 if,if...else 和 switch 语句来实现的。

16.1.1 if 语句

if 语句是 JavaScript 最常用的条件语句，就像 if 这个英文单词代表的意义那样，if 语句用来检查变量值是否满足条件，根据检查的情况再决定程序如何去做，if 语句通过 else 关键字实现更为复杂的条件判断和分支。

if 语句有如下三种形式。

- 单 if 形式：单 if 形式下省略了 else 关键字，程序执行时将首先判断逻辑表达式的值是否为 true（真），如果为 true，则执行花括号中语句块的内容，否则跳过花括号内容继续往下执行：

```
if (逻辑表达式){
...语句块...
}
```

- if...else 形式：程序执行时先判断逻辑表达式的值是否为 true，如果为 true，则执行语句块 1 的内容，否则执行语句块 2 的内容：

```
if (逻辑表达式){
...语句块 1...
```

```
}
else{
   ...语句块 2...
}
```

- if...else if 形式：程序执行时先判断逻辑表达式 1 的值是否为 true，如果为 true，则执行语句块 1 的内容，否则判断逻辑表达式 2 的内容，如果为 true，则执行语句块 2 的内容，否则跳到下一个 else if 语句继续判断，直到遇到 else 语句，则执行语句块 *n* 中的内容，如果前面有任何一个条件表达式为 true，则只执行响应语句块中的语句内容，其他语句块和语句块 *n* 中的代码都不会被执行。在这种形式中，最后的 else 语句也是可以省略的：

```
if (逻辑表达式 1){
...语句块 1...
}
else if (逻辑表达式 2){
   ...语句块 2...
}
...//还可以有多个 else if 语句
else{
   ...语句块 n...
}
```

当 if 语句后面只有一行需要执行的语句时，可以省略掉花括号。

下面是一个最基本的 if 语句：

```
if (a==1) console.log("hello");
```

连续的两个等于号（==）是 JavaScript 的等于逻辑判断符，上面这条语句检查了一个变量 a 的值是否为 1，如果为 1，则在调试控制台输出消息，否则不做任何动作。

16.1.2　条件运算符

我们在 if 语句里看到的用来判断条件是否成立的逻辑表达式叫作条件表达式。

条件表达式通常包括两个用于比较的值（前例中的值为 a 和 1）。这些值可以是变量、常数，甚至可以是表达式。

被比较的两个值中间的运算符叫作条件运算符，例如我们在前例中看到的“==”符号。条件运算符告诉 JavaScript 如何比较两个值。如，“==”运算符用来判断两个值是否相等。JavaScript 的条件运算符有如下几种：

- == 等于；
- != 不等于；
- < 小于；

- > 大于；
- >= 大于或等于；
- <= 小于或等于；
- === 严格等于（比较时不会发生类型转换）；
- !== 严格不等于（比较时不会发生类型转换）。

16.1.3　逻辑操作符

在我们需要依靠一个逻辑组合条件来判断进行怎样的操作时，可以使用逻辑操作符，逻辑操作符用于操作多于一个的布尔型变量和常量的组合，合理运用逻辑操作符可以大幅度地精简 if 语句的代码量。

JavaScript 的逻辑操作符主要有以下三个：

- && 逻辑与，必须两个布尔值都为 true 才会返回 true 值，否则返回 false；
- || 逻辑或，只要两个布尔值有一个为 true 就返回 true，除非两个都为 false 才会返回 false；
- ! 逻辑非，逻辑为 true 时返回 false，逻辑为 false 时返回 true，进行求逻辑反的运算。

比如我们要通知年龄超过 13 岁但是低于 29 岁的人领取团员证，就可以使用如下的代码：

```
if (age > 13 && age < 29) {
  console.log("请领取团员证");
}
```

16.1.4　switch 语句

如果我们要对同一个条件语句的不同可能性做出判断，可以用 if 语句来实现，也可以用一种更简洁的 switch 语句来实现。

switch 语句与 if 语句功能类似，不同之处是 switch 函数里表达式的值是一个变量，有可能是字符串、数值或者其他类型的数据。

```
switch(表达式){
case 值 1:
  …语句块 1…
  break;
case 值 2:
…语句块 2…
  break;
  default:
…语句块 n…
}
```



switch 语句执行时会将表达式的内容与 case 语句引用的值做对比，如果严格相等（===），则执行响应 case 语句下的代码快，否则跳过，如果所有的 case 语句引用的值都不能匹配，会执行 default 下的语句块。

需要注意的是，switch 语句的表达式一旦满足了与某个 case 语句引用值的匹配，就不再理会后面的 case 语句引用值的情况而一直执行下去。所以如果我们执行完一个语句块的内容需要 switch 语句跳出下面的判断时，就需要使用 break 语句，break 语句用于语句块的跳出，关于 break 语句更多的细节，我们会在下面的课程里详细讲解。

default 关键词用来定义所有 case 值都不能匹配的情况下程序需要执行的语句块。

下面是 switch 语句的一个示例：

```
switch (score){
  case 'A': console.log("优秀");
    break;
  case 'B': console.log("良好");
    break;
  case 'C': console.log("中等");
    break;
  case 'D': console.log("一般");
    break;
  case 'E': console.log("很差");
    break;
default : console.log("你的成绩搞错了");
}
```

这段程序会判断成绩的值，然后给予一个评价，如果所有的值都不符合，会提示成绩出错了。

16.2　循环语句

if 语句是为实现程序的分支结构服务的，我们经常会需要让程序重复操作某种功能，就需要使用 JavaScript 用来控制代码循环执行的流程控制语句。

16.2.1　for 循环

当一部分代码需要多次运行，并且执行次数是固定值的情况下，我们可以使用 for 循环语句来实现，for 循环是一种非常简洁的循环语句。

for 循环也叫计数循环，关键字 for 是创建循环的一种工具。for 循环使用一个变量记录循环执行的次数，这个变量叫作循环计数器，当计数器达到指定值时，循环停止。语句结构如下：

```
for (表达式 1;表达式 2;表达式 3){
...语句块...
}
```

表达式 1 指定一个变量并赋一个初始值。它被称为初始表达式，它设定了循环的初始状态。

表达式 2 是一个条件表达式，条件为 true 时循环才会继续，它被称作循环的条件。

表达式 3 是一个语句，循环的每次迭代都会执行它，通常用来让计数器的值递增，它被称作增量表达式，每次循环迭代的末尾会运行增量表达式。

下面是 for 循环的一个示例代码：

```
for (i = 1; i < 5; i ++ ){
   console. log(i + "\n");
}
console. log("循环体结束")
```

代码开始执行时变量 i 会被赋初值 1，每次循环结束变量 i 的值会加 1，直到 i 的值不小于 5 则循环停止，所以这个例程里面的循环语句块会被执行 4 次。

16.2.2　for in 循环

使用关键字 in 的 for 循环控制语句也被叫作 for in 循环，语法格式如下：

```
for (索引变量 in 对象){
   ...语句块...
}
```

for in 循环主要被用来遍历数组中的元素和遍历 JavaScript 对象的属性，其计数器也就是数组元素的索引值。例如：

```
vardays = [ "Tuesday" , "Wednesday" , "Thursday" , "Friday" , "Saturday" , "Sunday" ];
for (index in days){
console. log("索引号" + index + ":" + days[ index] + "\n");
}
```

代码执行后将会把数组中的每一个元素列举出来。

16.2.3　while 循环

JavaScript 关键字 while 也可以发起循环。与 for 循环不同，while 循环不计数，它会判断一个条件表达式的值，只要为 true 就会一直执行循环，所以 while 循环也叫作条件循环，同样属于条件循环的还有我们后面要讲到的 do while 循环。



while 循环的语法与 if 语句非常相似：

```
while(条件表达式){
  ...语句块...
}
```

程序执行后开始判断条件表达式的值，如果为 true 则进入循环，直到条件表达式不成立时退出循环。

下面是一个简单的 while 循环示例：

```
while (i<5){
console.log(i+"\n");
  i++;
}
```

该循环使用了一个计数器 i 在循环中迭代，当 i 值到 5 时停止循环。

16.2.4　do while 循环

do while 循环与 while 循环类似，唯一的不同在于：do while 语句在循环的末尾检查条件表达式，而 while 语句是在循环的开头检查的。

do while 循环的语法：

```
do{
...语句块...
}
while (条件表达式);
```

程序执行时先执行循环体语句块的内容，执行完毕后判断循环条件，如果条件为 true 则返回语句块继续执行，否则跳出循环。

do while 循环在循环的末尾检查循环条件，所以即使条件永远不满足，循环中的语句也会至少执行一次。

例如下面的 do while 循环：

```
do{
  console.log(i+"\n");
  i++;
}
while (i<5);
```

16.2.5　break 退出循环语句

在循环中遇到 break 语句时，程序会立即跳出循环体，如下面的示例：

```
while (i<10){
console.log(i+"\n");
  i++;
if (i>6) break;
}
```

每次循环时都会判断变量 i 的值是否大于 6，如果大于 6 则马上跳出循环体，终止循环语句的执行。

除在循环语句中有应用外，break 也用于条件语句的跳出判断，这方面的内容我们在前面 switch 条件语句的学习中曾经接触过。

16.2.6　continue 继续循环语句

continue 是用来跳过循环体其余部分的语句，与 break 不同，它不会让循环终止，而是让程序跳出本次循环，进入下一次循环。例如：

```
while (i<10){
console.log(i+"\n");
  i++;
if (i>6) cntinue;
  console.log("这是一个完整的循环");
}
```



课后作业

1. 使用 switch 语句实现一个根据系统日期显示今天是星期几的小程序，并将结果打印在屏幕上。

2. 使用 for 循环语句实现在屏幕上打印一个九九乘法表和倒三角的图形。

第7部分
JavaScript 进阶

通过上一章的学习我们了解了 JavaScript 语言在面向过程的程序设计上的一些特性，同时，JavaScript 作为一种成熟度非常高的编程语言，也提供了面向对象编程的能力。

面向对象编程是现代编程思想中的重要理念，它让编程的思维方式更加趋向于我们对现实世界的认识，它可以帮助我们编写逻辑清晰的可以复用的代码。

本章我们将学习 JavaScript 语言中有关面向对象编程的技术，以及 JavaScript 的一些新特性和更加简洁的编写 JavaScript 代码的方式。



本章要点

- 面向对象编程（OOP）
- 类与对象
- 函数
- JSON 语法
- ES6 新语法特性

第 17 天

面向对象程序设计（OOP）

17.1 类和对象

面向对象编程（Object Oriented Programming，OOP）是相对于面向“过程”编程来说的。

我们在上一章学习的就是面向过程的编程方法：把数据保存在变量里，通过指令操作变量，一步步地完成程序的执行过程。我们可以去创建、修改和删除变量里保存的数据，数据和程序是分离的，这种程序设计方式叫作面向“过程”的程序设计。

在面向对象编程的方式中，数据打包在一个称作“对象”的独立容器里，每个对象都有自己的属性（数据）可以被配置，都有自己的方法（指令）可以被操作。

随着程序的复杂化，我们需要使用一些编码技术帮助我们保持对代码的掌控，面向对象的程序设计应运而生，面向对象编程可以帮助我们编写清晰可靠的可以复用的代码。

面向对象是把对现实世界的认识和抽象的方法引入到程序设计中，它是基于“类”与“对象”的概念的，类和对象模拟了我们对真实世界的认识。

类（class）也就是种类，而对象（object）是类的实体体现，类是对象的模板。

类的实质是一种数据类型；就像数值型、字符串等基本数据类型，类是一种更为复杂的数据类型。一个类，有可能是一个包含很多种简单或复杂的数据类型的集合。因为类的本质是数据类型，并不是真实存在的数据，所以类并不会实际存在于内存中，也不能被赋值或赋值给变量。

就像变量是数据类型的实例一样，类的实例就是对象，我们不能直接操作类，只有类被实例化为对象时，才能被操作。

实例（instance）也就是“实体的例子”，类只是一种类型规范，它是抽象的，没有实体，对象就是类的实例，是具象的，所有类的对象都是实例。

听起来可能难以理解，我们打一个比方，“狮子”是个类型，“狮子”的概念是抽象的，我们对狮子的认识是一种猫科动物、大型、食肉、群居，狮子可以捕食、休息、卖萌，这些都是对“狮子”这个类型的定义，而具体到一个狮子个体，比如有一只叫作“艾尔莎”的狮子，就是个真实的个体了，是具象的，艾尔莎这只狮子是我们真正可以去观察、去了解的个体。



在这个例子里，“狮子”类型就是一个类（class），狮子“Eric”就是类的一个实例（instance）：对象（object）；“猫科动物、大型、食肉、群居”这些描述就是对象的“属性”（property）；Eric 的“捕食、休息、卖萌”就是对象的“方法”（method）。

类、对象、属性、方法都是面向对象编程中最重要的概念，深入理解这些概念是我们学好面向对象编程的前提。

面向对象的编程方法具有四个基本特征：

- 抽象；
- 继承；
- 封装；
- 多态性。

抽象就是从实例中抽取与目标有关的方面，忽略无关的方面。

比如我们要设计一个员工福利管理系统，考察一个员工对象时，我们只关心他的部门、职位、福利种类、福利金额等内容，而不用去关心他的政治面貌、学历这些与福利无关的信息。

继承是一种连接类的层次结构模型，一个新类可以从现有的类中派生，这个过程称为类的继承。新类继承了原始类的特性，称为原始类的子类，而原始类称为新类的基类(父类)。

封装是类和对象的主要特性，封装就是把过程和数据包装起来，对数据的访问只能通过已被定义的接口。

多态性是指允许不同类的对象对同一个消息做出响应。

面向对象的程序设计相比面向过程有如下的优点：

- 代码可复用，设计时可以大量重用现有的已经调试成熟的代码，开发周期更短，效率高更，可靠性更高。
- 改变程序的功能只需要修改响应的局部模块就可以了，所以应用程序更加易于维护和升级迭代。继承和封装特性让代码修改带来的影响更局部化，安全风险更低。

JavaScript 并不是一种严格面向对象的程序设计语言。在纯粹的面向对象编程语言里，最基本的程序单位是类，类与类之间提供严格的继承关系。而 JavaScript 并没有提供规范的语法让开发者定义类，JavaScript 的类和对象通常都是通过变量与函数实现的。但 JavaScript 对函数的操作提供了强大的针对对象特性的支持，让我们可以编写实用的面向对象代码，提高代码编写和执行的效率。

JavaScript 语言的对象就是为了实现某种功能而组织的一系列变量与函数的集合，对象中的变量就是对象的属性，对象中的函数就是对象的方法。

对象访问属性和方法都是通过圆点符号（.）实现的。例如，我们经常用到的打印语句，就是对对象方法的操作：

```
console.log("微信:2264538870");
```

JavaScript 是基于对象的脚本语言，它提供了丰富的内置“类”来让我们定义对象，这些内置类包括：

- String：字符串类；
- Number：数值类；

- Date：日期类；
- Math：数学类，包含大量的有关数学运算的方法；
- Object：对象类；
- Function：函数类；
- Array：数组类；
- Error：错误累。

17.2　创建对象

JavaScript 提供了多种创建对象的方式，包括用内置类创建“直接实例”的方式、构造函数的方式，后面我们还会学到更简洁的用 JSON 语法创建对象的方式。

17.2.1　直接创建 Object 类的对象实例

前面我们知道了 JavaScript 有一个内置类叫 Object 对象类，在 JavaScript 里所有的对象都是 Object 类的子类，所以我们可以直接用它来创建对象关键字创造一个空白的对象：

```
//用 Object 类创建一个对象
var myObject = new Object( );
```

这样就得到了一个叫作 myObject 的空对象，空对象不包含任何属性和方法，JavaScript 是一种动态语言，我们可以随时动态地为对象添加属性和方法：

```
myObject. name = " 老王。";
myObject. age = 28;
```

这样，对象就有了两个属性：name 和 age。

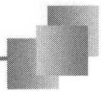
这种列表赋值的语法你有没有觉得特别熟悉？是不是和数组的赋值方式完全一样？没错，JavaScript 对象的本质就是一个关联数组，我们也可以用操作数组的语法来访问对象的属性，如用 for in 函数遍历对象的属性内容。

关联数组是一种具有特殊索引方式的数组。我们不仅可以通过非负整数来索引它，还可以使用属性名来索引它。

关联数组的元素没有特定的顺序，你可以把它们想象为一组卡片。每张卡片左边是索引而右边是数值。

给对象添加方法需要首先定义一个函数，然后把它附加到对象里作为方法，如下面的代码，定义了 newFunc( ) 函数，之后通过赋值语句把它附加到 myObject 对象里作为 listText 方法：

```
function newFunc( ){
```



```
    console.log("学编程从小程序开始很简单");
};
myObject.listText = newFunc;
```

17.2.2　用 new 关键字创建对象

用 new 关键字创建对象的方法也叫 new 构造函数法，示例如下：

```
var person = new Object();
person.name = "老王";
person.age = 28;
```

这种方式使用 new 关键字，接着跟上 Object 构造函数，再来给对象实例动态添加上不同的属性，相对来说比较烦琐。

构造函数主要用来在创建对象时初始化对象，即为对象成员变量赋初始值，构造函数一般与 new 关键字配合使用创建对象和定义对象的属性。

17.2.3　this 关键字

在 JavaScript 语言里，this 关键字用来指向函数的上一级对象（父对象）。

例如，我们把函数 newFunc()赋值给了 myObject 对象的一个方法 changeName：

```
myObject.change = myFunc;
```

当我们在 changeName()方法中应用关键字 this 时，指向的就是包含它的对象 myObject。this 关键字只能应用在函数内部，它指向的永远都是调用这个函数的对象。

17.3　使用对象

对象的使用方法是这样的：

```
myObject.listText();
```

要注意的是，我们在把函数关联到对象时，只使用函数名称，不包括括号，因为如果有括号，会变成一条赋值语句，执行把函数值赋予对象的属性的操作。

我们也可以采用匿名函数的方式设置对象方法，在这种方法中不需要给函数命名，如上面的例程，我们可以修改为这种形式：

```
myObject.listText = function(){
    console.log(this.text);
}
```

17.4　继承

前面我们学过，面向对象的程序设计语言有一个很重要的特点，就是继承。

继承是指从一个已经存在的类创建另一种类的过程，一个类可以继承另外一个类的所有属性和方法，继承的类叫作子类，被继承的类叫作父类，子类可以在父类的基础上扩展属性和方法。

JavaScript 语言支持类和对象的概念，但是并没有明确的继承的概念，我们可以用 prototype 关键字模拟实现类和对象的继承。

我们先来定义一个对象：

```
function someOne( ) {
  this. id = " " ;
  this. name = " " ;
  this. info = function( ) {              //定义了对象的一个方法
    console. log( this. id) ;
    console. log( this. name) ;
  }
}
```

someOne 对象有两个属性：id 和姓名，和一个把这些属性显示在控制台的方法。

如果我们要为 someOne 里面的男性再创建一个 male 对象，就可以让新建的对象从 someOne 对象里继承它的属性和方法，并且增加新的属性 sexual 和显示性别的方法 showSexual。

首先，我们创建 male 对象并定义它的属性和方法：

```
functionmale( ) {
this. sexual = " " ;
this. showSexual = function( ) {
console. log( this. sexual) ;
  }
}
```

最后一步就是让 male 对象从 someOne 对象里继承属性和方法：

```
male. prototype = new someOne( ) ;
```

任务完成，可见，用 prototype 关键词实现对象继承的语法如下：

```
子对象名 . prototype = new 父对象名( ) ;
```



课后作业

1. 用 Object 对象类写一个对象实例，对象包括三个属性：学号，年龄和姓名，再创建一个方法，把序号、姓名、性别数据输出到控制台。

2. 在上题的基础上创建一个子类，增加年级属性和把年级属性输出到控制台的方法。

第18天

函　数

18.1　定义函数

程序在执行的过程中我们经常会遇到要重复去处理完全相同或者非常相似的任务，为了提高代码的编写和执行效率，我们可以把这些功能重复的代码用模块化的形式包装起来，每次执行同样的任务时，只要用一两行语句调用这个模块就可以了，JavaScript 称这种被我们打包起来处理特定任务的代码模块叫作“函数”。

函数在 JavaScript 语言里拥有非常高的地位，在前面的学习中我们已经了解到，JavaScript 的类是基于函数的，对象是基于函数的，方法也是基于函数的，可以说整个 JavaScript 语言的面向对象编程都是基于函数的。

微信小程序的程序入口就是一个函数，每一个页面的逻辑代码也是从函数引入的，我们在微信小程序上的每一个操作都与函数有着密不可分的关系。

函数可以让程序的其他部分调用它，使用函数可以让代码更加易读，更加容易维护，它最大的作用就是提供代码复用的能力。

JavaScript 的函数功能非常丰富，它是 JavaScript 的基本组成模块。

18.1.1　定义命名函数

命名函数也就是有函数名的函数，是相对于“匿名函数”（没有名称的函数）来讲的，JavaScript 创建命名函数使用 function 命令，语法规则如下：

```
function 函数名(参数列表){
...函数体语句块...
}
```

下面是一个定义函数的例子，定义了一个 sayHi( ) 函数：

```
function sayHi( ){
  text = "大家好!";
```



```
    console. log(text);
}
```

关键字 function 用来表明我们要定义一个函数，后面的 sayHi( )是函数的名称，花括号里面是构成函数体的语句块，这个函数的作用是在调试控制台里打印一个字符串“大家好!”。

需要注意的是，关键字 function 必须小写的函数的名称和变量名称一样区分大小写。

函数在定义时不会被加载执行，只有在被调用时才会被执行。我们可以在需要执行函数的地方直接用函数名加参数的方式调用这个函数：

```
sayHi( );
```

定义函数的语法非常简单，我们在定义函数的同时也获得了一个同名的对象。

18.1.2　定义匿名函数

JavaScript 语言还提供一种定义匿名函数的方式，让函数的定义和使用变得更加简单。

匿名函数也就是说被定义的函数没有函数名，创建匿名函数的语法规则如下：

```
function (参数列表){
...函数体语句块...
}
```

定义匿名函数时不需要指定函数名，函数的参数列表紧跟着 function 关键字。需要注意的是，用于包装函数体语句块的花括号后面要加上分号（;）结尾。

有些朋友可能要问，上一节我们学习的有名字的函数可以用函数名来调用函数，匿名函数怎样调用呢?

匿名函数的调用比命名函数还要简单，我们在定义时就可以把整个匿名函数赋值给一个变量，调用函数时只要使用这个变量就可以了，函数赋值给变量的方法见下面的例子：

```
varnewFunc = function(userName){
console. log("这是一个匿名函数的赋值实例");
console. log("您是用户" + "userName" + "吗?");
};
newFunc("老王");
```

这段代码用 function 方法定义了一个匿名函数，并且把函数赋值给了变量 newFunc，最后一行用变量名调用了这个匿名函数。

匿名函数为函数的定义和使用提供了更简洁的方法，当我们需要为对象定义方法时，使用匿名函数能让程序看起来更简洁，更直观。

18.1.3　用 Function 类创建匿名函数

在上一节我们了解到，JavaScript 提供了一个 Function 内置类，JavaScript 语言定义的所有函数都是 Function 类的子类，所以我们可以用与上一节创建对象类似的方式来创建匿名函数：

```
varnewFunc = newFunction('userName'," console. log("这是一个 Function 内置类定义的函数");"
+console. log("您是用户" +"userName" +"吗?"););
newFunc("老王");
```

这段代码与上例的功能相同，定义了相同内容的匿名函数并赋值给了 newFunc 变量。

在用 Function 类构造函数的方法中，Function 可以接收一组字符串形式的参数，其中最后一个字符串参数是函数体语句块，每条语句用分号（;）结尾，语句间用字符串的连接符（+）连接。

我们看到采用这种方法定义匿名函数时，当函数体语句块内容较多时，整个定义的过程会让代码的可读性变得非常差，内容混乱不堪，所以不建议用这种方法定义函数。

18.2　函数的参数

我们可以通过向函数传递不同的数据的方法让同一个函数执行不同的功能，实现的方法是在调用函数时给它传递“参数”，一个函数可以传递一个或者几个参数，如下面是一个函数的定义：

```
function price(x){
console. log(x * 5);
}
```

现在我们调用这个函数：

```
price(2);
```

控制台窗口会显示计算的结果：10。

我们也可以把一个变量作为参数传递给函数，例如：

```
var num =20;
price(num);
```

也可以把几个参数传递给函数，让函数负责数值的处理，几个参数之间要用逗号分割，例如：

```
function min(h,x){
```

```
    console. log(h * 60 + x);
}
min(12,20);
```

JavaScript 的函数可以根据需要，定义任意多个可供传递的参数。

18.3　函数的返回值

函数的返回值，也就是函数被调用后返回给调用者的结果，返回的既可以是数值，也可以是 JavaScript 支持的其他一种数据类型，如字符串型或者布尔型等。

函数的返回值利用 return 语句实现，比如上文的例程，我们可以改为下面的方式，让函数返回一个数据给调用它的程序：

```
function min(h,x){
    return h * 60 + x;
}
var minute = min(3,15);
```

函数的计算结果就会赋值给变量 minute 了。

课后作业

1. 思考一下，我们创建了两个函数，函数 A 和函数 B，函数 B 本身能不能作为参数存在于函数 A 中呢？

2. 在你的开发环境中实现上面的案例，并观察其效果。

第19天

JSON语法详解

19.1 JSON的语法结构

JSON读作Jason，是JavaScript Object Notation的缩写，翻译成中文就是JavaScript对象表示法，它是JavaScript语言提供的一种轻量级的数据交换格式，基于文本，它的作用类似XML语言。

JSON是JavaScript原生支持的语法格式，这意味着在JavaScript中处理JSON数据不需要任何额外的支持，如API或SDK工具包之类。

JSON语法是JavaScript对语法的子集，JSON语法的核心价值就在于，它可以把复杂的定义对象的方法转化为简单的操作字符串的形式。

我们知道小程序的配置文件是基于JSON语法的，同时JSON语法也能给小程序的逻辑层代码增加更为强大的功能。

JSON有两种表示结构，分别用于对象的操作和数组数组的操作。

对象结构的JSON语句块与JavaScript语句块一样，包含在花括号内。语句块由一条或者多条JSON语句构成。

JSON语句由“key（关键字名称）/value（值）”构成，我们管这种结构叫作“名称、值对”，关键字名称与值之间用冒号（:）分割，JSON语句之间用逗号（,）分割。例如：

```
{
    key1:value1,key2:value2,...
}
```

“名称、值对”中的关键字名称是字符串的形式，而值的类型可以是字符串、数值、布尔型数据、null，甚至可以是对象或者数组。

以下是一个基本的名称、值对：

```
{"name":"老王"}
```

JSON的数组结构以方括号开始和结束，中间由花括号括起来的元素组合组成，例如：

```
[
    {
        key1:value1, key2:value2,
    },
    {
        key3:value3, key4:value4
    }
]
```

19.2　使用 JSON

作为 JavaScript 的语法子集，JSON 创建的对象和数组可以方便地在 JavaScript 中读写。读写 JSON 对象有两种方法，分别是利用点号（.）操作符和“[key]”的方式。

我们首先定义一个 JSON 对象，代码如下：

```
var obj = {
  1 : "value1",
  "2" : "value2",
  count : 3,
  person : [ //数组结构 JSON 对象,可以嵌套使用
             {id : 1,name : "老王"},
             {id : 2,name : "老张"}
           ],
    object :{ //对象结构 JSON 对象
             id : 1,msg: "对象里的对象"
           }
};
```

19.2.1　从 JSON 中读数据

从 JSON 中读数据，代码如下：

```
function ReadJSON(){
        alert(obj. 1);                  //会报语法错误,可以用 alert(obj["1"]);说明数
                                          字最好不要做关键字
        alert(obj. 2);                  //同上

        alert(obj. person[0]. name);    //或者 alert(obj. person[0]["name"])
```

```
        alert(obj. object. msg);                    //或者 alert(obj. object["msg"])
    }
```

19.2.2　向 JSON 中写数据

比如要往 JSON 中增加一条数据，代码如下：

```
function Add(){
        //往 JSON 对象中增加了一条记录
        obj. sex = "男" //或者 obj["sex"] = "男"
    }
```

19.2.3　修改 JSON 中的数据

我们现在要修改 JSON 中 count 的值，代码如下：

```
function Update(){
        obj. count = 10; //或 obj["count"] = 10
    }
```

19.2.4　删除 JSON 中的数据

我们现在实现从 JSON 中删除 count 这条数据，代码如下：

```
function Delete(){
        delete obj. count;
    }
```

可以看到 count 已经从 JSON 对象中被删除了。

19.2.5　遍历 JSON 对象

可以使用 for...in...循环来遍历 JSON 对象中的数据，比如我们要遍历输出 obj 对象的值，代码如下：

```
function Traversal(){
        for (var c in obj){
            console. log(c + ":", obj[c]);
        }
    }
```



课后作业

1. 将上面的案例在你的开发环境中实现一遍，并在运行环境中观察其效果。

2. 前面我们分别学习了 XML 和 JSON 语法的知识，这是在互联网中流行的两种用来做数据传输用途的语言，说说你理解的两种语言的优缺点。

第20天

ES6 语法

20.1 ES6 简介

ES6 是 ECMAScript 6 的简称，也就是 ECMA 标准化组织颁布的 JavaScript 语言的新标准。ES6 是在 2015 年发布的，所以 ES6 又被称作 ECMAScript 2015。

目前我们使用的 JavaScript 语法标准是 ES5，也就是 ECMA 颁布的第五代语法标准。

我们在前面的学习中已经知道，JavaScript 的语法并不支持直接定义类（class），而 ES6 语法更接近传统面向对象语言，我们可以在 ES6 里通过 class 关键字来定义类。

虽然 ES6 的绝大部分功能 ES5 都可以做到，但 ES6 的出现让我们可以用更简洁、更清晰的语法来实现这些功能。

现在并不是所有浏览器都支持 ES6 特性，但是微信小程序支持 ES6 语法转译，开关在小程序开发工具的项目设置页面里，选中“开启 ES6 转 ES5”就能让你用 ES6 语法为小程序编写逻辑代码了。

ES6 在 ES5 的基础上增加了很多新特性，更新的范围很大，但是常用的特性并不多，本节我们将对这些常用的特性做一个分析。

下面我们介绍 ES6 中最常用的一些语法关键词 let、const、class、extends、super 和 arrow functions 箭头函数，它们分别用于变量的定义、类的操作和函数的定义，可以让我们更高效地完成代码的编写。

20.2 ES6 常用语法详解

20.2.1 let 变量声明语句

let、const 关键字都用作变量的声明，与 var 关键词类似，不同点在于 let 定义的变量是局部变量，作用域只在 let 语句所在的语句块内。例如：

```
let name = 'zach '
```



```
while(true){
    let name = 'obama'
    console.log(name)   //obama
    break
}
console.log(name)       //zach
```

20.2.2　const 常量声明语句

与 var 和 let 都不同，const 用来声明常量，也就是被声明的值在程序的生命周期中不能被改变。例如：

```
constPi = 3.1415926;
```

如果我们试图去改变一个被 const 关键字声明的常量，语法检查系统就会报错。

20.2.3　类操作语句

ES6 引入了类（class）的概念，用关键词 class 定义类，让描述对象的语法与真正面向对象编程的语言更加接近，可读性更强。例如：

```
class humen{
    constructor(){
        this.type = 'humen'
    }
goto(posi){
        console.log(this.type + 'goto ' + posi)
    }
}
let humen = newhumen();
humen.goto('Beijing')     //运行结果 humen goto Beijing
```

上例用 class 关键字定义了一个名为 humen 的类，里面的 constructor 方法是构造方法，this 关键字代表实例对象。constructor 语句块内定义的方法和属性是对象实例本身的，而 constructor 语句块外定义的是实例对象可以共享的属性和方法。

我们知道，ES 的继承机制是通过 prototype 关键字来模拟实现的，多次定义，语法烦琐，容易出错。ES6 中类的继承是通过 extends 关键字来实现的，语法简单有力，清晰明了。例如，下面的例子定义了一个 male 类，这个类通过 extends 关键字继承了 humen 类的所有属性和方法：

```
class male extends humen{
    constructor(){
```

```
        super()
        this. type = 'male '
            }
        }
        let male = newmale()
        male. goto('Shanghai ')//运行结果 malegotoShanghai
```

在这个例子中，我们还用了 super 关键字指代当前类父类的实例（也就是父类的 this 对象），这也是 ES6 的新增用法。

子类调用 this 方法时需要注意，必须在 constructor 方法中先调用 super 方法。这是因为子类的 this 对象是继承自父类的，所以必须先调用 super 方法创建出父类的 this 实例对象，否则子类就找不到自己的 this 对象。

ES6 的继承机制，实质是先创造父类的实例对象 this（所以必须先调用 super 方法），然后再用子类的构造函数修改 this。

20.2.4　arrow function 箭头函数

arrow function 箭头函数（ => ）的应用让 ES6 的代码编写效率获得了极大的提升，这也是 ES6 最常用的一个新特性了，用它来定义函数要比 ES5 语法简洁清晰得多：

```
function(value){return value * 20;}    //ES5 的语法
(value) => value * 20                  //ES6
```

是不是要比传统的语法简单很多？

复杂的语句需要用花括号（{}）把代码包裹起来：

```
function(num,price){
num = num + 5;
price = price * 2;
    return num * price;
}
(num,price) => {num = num + 5;price = price * 2;return num * price};
```



课后作业

1. 用 ES6 语法定义一个常量，值是数学常数 PI 的平方。

2. 用 ES6 语法定义写一个类，对象包括两个属性：学号和姓名，再创建一个子类，增加性别属性，并增加把三个属性输出到控制台的方法。

第8部分
工具和社区

微信小程序经历了内测和公测，发布了集代码编写、调试和程序发布为一体的IDE，目前的成熟度已经很高，但是基于代码编写的UI设计仍然会让人感觉烦琐和不友好。

本章我们介绍一些已经上线的第三方开发的编程工具，方便我们像写Office文档一样完成小程序的前端开发。另外，我们还会介绍一些初具规模的小程序学习社区，在社区里我们可以交流、沟通和分享我们的学习心得和编程体会，一个分享的环境对学习，对工作，都是非常重要的。



本章要点

- 前端可视化工具Coolsite360
- 即速应用
- 学习社区

第21天

开发工具和学习社区

21.1 Coolsite360 可视化工具

Coolsite360 是第一款针对微信小程序的可视化设计工具，于 2016 年 11 月 4 日上线发布公测，也就是在微信小程序提起公测一个月左右的时间就上线了，不得不佩服他们的速度。

Coolsite360 集成了微信目前发布的所有小程序组件，我们可以抛开 WXML 和 WXSS 烦琐的不直观的设计细节，在 Coolsite360 工具里通过类似编辑 word 文档的方式进行小程序的 UI 设计，降低前端的开发难度，让几乎每个人都能设计小程序的前端页面。Coolsite360 的用户界面如图 21-1 所示。

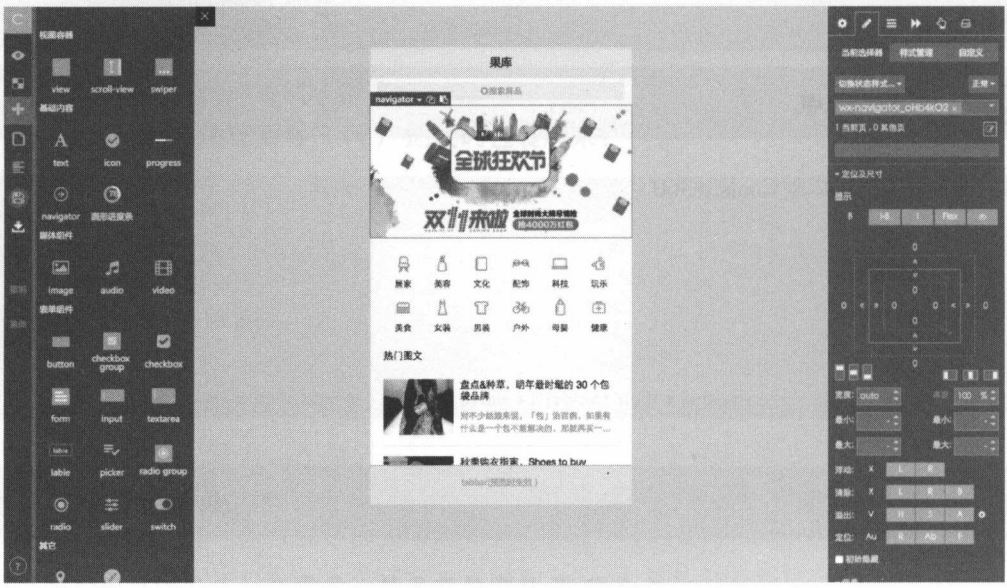


图 21-1　Coolsite360 的用户界面

Coolsite360 支持 Flex 布局方式，可以直接导出符合小程序规范的代码文档。它具有下面的特点：

- 集成的组件全面；

- 支持 Flex 布局模式和 rem 尺寸单位；
- 原生支持腾讯的 WeUI 设计规范；
- 内置模板和布局样式组件；
- 支持组件的扩展；
- 支持用户交互开发。

Coolsite360 集成了微信定义的所有小程序可视组件，方便设计师直接以可视化的方式添加使用。Coolsite360 的组件面板如图 21-2 所示。

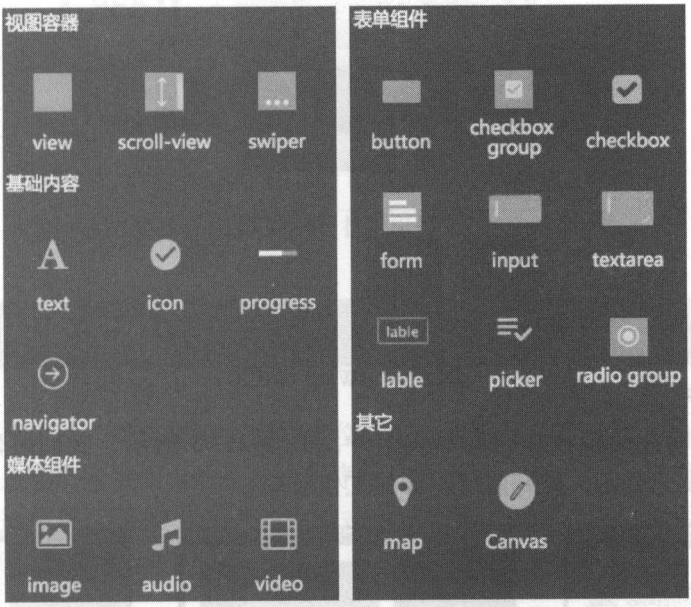


图 21-2　Coolsite360 的组件面板

它也支持强大的 Flex 布局方式，组件尺寸使用 rem 单位等，如图 21-3 所示，让小程序的界面能与设备屏幕更好地适配，但是目前还不支持微信的 rpx 尺寸单位，据 Coolsite360 的开发人员透露，近期会升级对 rpx 的支持。

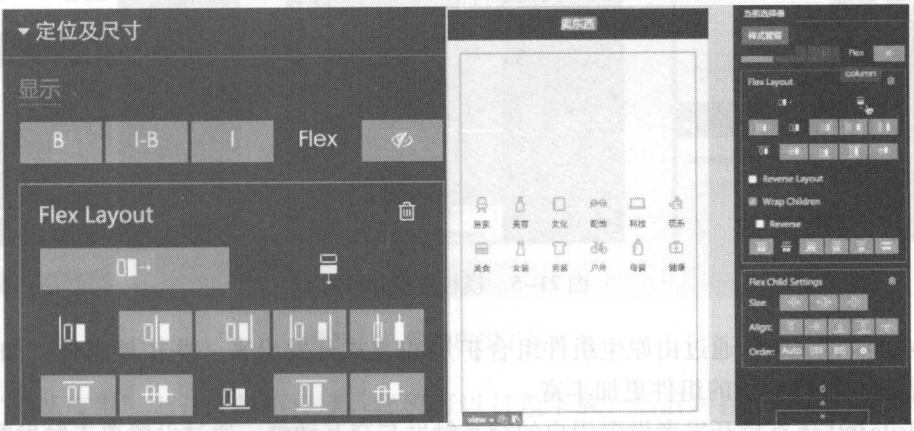


图 21-3　布局和尺寸定义面板

Coolsite360 支持微信官方提出的小程序设计规范要求，通过内置的 WeUI 样式库，如图 21-4 所示，能够帮助我们快速完成兼容 WeUI 设计规范的前端设计工作。

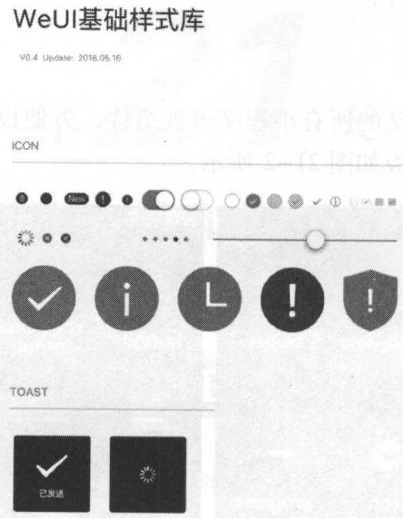


图 21-4　WeUI 样式库

WeUI 内置也写应用模板和布局样式组件，如图 21-5 所示，我们可以直接套用修改，通过拼装的方式设计页面，提升前端设计的工作效率。



图 21-5　模板和组件样式

同时，Coolsite360 通过由原生组件组合扩展的方式开发出来一些扩展组件，如图 21-6 所示，让我们可以应用的组件更加丰富。

Coolsite360 还支持开发者设置用户的交互触发与交互动画，通过设置单击触发配合组件的动画，就能做出侧滑、弹出蒙层等交互效果。



图 21-6　扩展组件

21.2　即速应用

与 Coolsite360 类似，“即速应用”同样定位于微信小程序的可视化页面设计和编程。

“即速应用”是咫尺网络推出的一款小程序第三方设计工具，与原型设计工具墨刀的操作方式类似，“即速应用”采用拖曳组件的方式生成页面。如图 21-7 所示为即速应用工具页面。



图 21-7　即速应用工具页面

在“即速应用”上开发的小程序，代码可以打包下载，然后对接到小程序的开发工具里。这样做省去了很多技术上的麻烦，为开发者提供了便利。专业的程序员还可以进行二次开发，实现更多高级的功能。

“即速应用”最大的亮点是，用它开发出来的可以在 PC 端浏览器端上直接预览，给我们提供了一种更为直观、方便调试小程序的可能。

“即速应用”拓展了小程序的后台管理功能，开发者在“即速应用”后台可以直接编辑修改小程序的数据，实现了动态数据绑定。对于电商等有大量数据需求的小程序开发者，实时的数据更新系统可以简化数据的更新和管理过程。如图 21-8 所示为后台数据管理页面。



图 21-8　后台数据管理页面

同样地，作为一款可视化的前端设计工具，即速应用也提供了大量的模板，可以直接套用修改。

21.3　开发社区介绍

1. 微信小程序俱乐部

网址：http://www.wxappclub.com

微信小程序俱乐部是一个基于原创作品的小程序技术开发和运营平台，如图 21-9 所示。

社区推出了一些优质的小程序开发和运营公开课程，目前已经有大量技术和运营人员聚集。

可以推荐的是该站小程序文档中心的手册文档，比较注重实战。

2. 青雀社区

网址：http://bbs.larkapp.com

青雀社区是白鹭时代旗下的开发者交流社区，以前主要是做 H5 方面的开发和讨论，目前开发了一款支持小程序的支持 IDE：wing，还有类似 Coolsite360 的小程序可视化生成工



图 21-9　微信小程序俱乐部

具：青雀平台，还有就是青雀论坛。

3. GetWeApp

网址：http://www. getweapp. com

GetWeApp 是一个学习微信小程序开发的社群，他们的理念是通过学习优秀开发者的开源代码，快速提高自身的开发水平。它的资源中心收集了开发过程中可能会需要的开源库、工具、网站等。



图 21-10　GetWeApp 界面

4. CSDN

网址：http://www. csdn. net

老牌的开发者社区，包括大量的小程序资源下载、课程试听、话题讨论等。

5. v2ex 技术区

网址：https://www.v2ex.com

v2ex 是汇集了各种有趣的话题和前沿信息的讨论区网站，它的技术区有关于微信小程序和其他 IT 技术的讨论。

6. 知晓程序

网址：https://minapp.com

知晓程序有一个小程序商店，在里面我们可以扫码体验别人发布的微信小程序，弥补了小程序没有 store 平台的缺憾，不过目前只能通过微信联系“知晓程序”管理员申请体验权限的方式体验小程序，小程序正式发布后会有改观。

7. 小程序开发论坛

网址：https://weappdev.com

另一个微信小程序开发技术的分享站点，首页扫码可以加入他们的讨论群，活跃度很高。

8. 蜂鸟微信小程序

网址：http://weixin.huosu.com

一个小程序 Demo 的收集发布站，也包括了免费教程和讨论社区等。

9. 小鹅通

网址：https://www.xiaoe-tech.com

提供微信小程序的定制服务，创始人都来自腾讯。

附录
微信小程序工具书

附录包括了微信官方提供的小程序组件详解和API接口详解，这部分内容来自于微信官方发布的开发文档，感谢微信小程序团队的辛勤工作，让我们在很短的时间内能够了解到微信小程序的几乎全貌，也让我们能够更快速地上手小程序开发。



附录内容

- 小程序组件详解
- 小程序 API 详解
- JavaScript 的关键字和操作符

附录 A

小程序组件详解

A.1 视图容器

A.1.1 view

1. view

视图容器组件如表 A-1 所示。

表 A-1 视图容器组件

属性名	类型	默认值	说明
hover	Boolean	false	是否启用单击态
hover-class	String	none	指定按下去的样式类。当 hover-class="none" 时，没有单击态效果
hover-start-time	Number	50	按住后多久出现单击态，单位毫秒
hover-stay-time	Number	400	手指松开后单击态保留时间，单位毫秒

2. 示例代码 A-1

```
<viewclass="section">
<viewclass="section__title">flex-direction:row</view>
<viewclass="flex-wrp" style="flex-direction:row;">
<viewclass="flex-item bc_green">1</view>
<viewclass="flex-item bc_red">2</view>
<viewclass="flex-item bc_blue">3</view>
</view>
</view>
<viewclass="section">
<viewclass="section__title">flex-direction:column</view>
```

```
<viewclass = "flex - wrp"  style = "height:300px;flex - direction:column;" >
<viewclass = "flex - item bc_green" >1 </view >
<viewclass = "flex - item bc_red" >2 </view >
<viewclass = "flex - item bc_blue" >3 </view >
</view >
</view >
```

A. 1. 2　scroll - view

1. scroll - view

可滚动视图区域组件如表 A-2 所示。

表 A-2　可滚动视图区域组件

属 性 名	类　型	默 认 值	说　明
scroll - x	Boolean	false	允许横向滚动
scroll - y	Boolean	false	允许纵向滚动
upper - threshold	Number	50	距顶部/左边多远时（单位 px），触发 scrolltoupper 事件
lower - threshold	Number	50	距底部/右边多远时（单位 px），触发 scrolltolower 事件
scroll - top	Number		设置竖向滚动条位置
scroll - left	Number		设置横向滚动条位置
scroll - into - view	String		值应为某子元素 id，则滚动到该元素，元素顶部对齐滚动区域顶部
bindscrolltoupper	EventHandle		滚动到顶部/左边，会触发 scrolltoupper 事件
bindscrolltolower	EventHandle		滚动到底部/右边，会触发 scrolltolower 事件
bindscroll	EventHandle		滚动时触发，event. detail = {scrollLeft, scrollTop, scrollHeight, scrollWidth, deltaX, deltaY}

使用竖向滚动时，需要给 < scroll - view/ > 一个固定高度，通过 WXSS 设置 height。

2. 示例代码 A-2

```
<viewclass = "section" >
<viewclass = "section__title" > vertical scroll </view >
<scroll - viewscroll - y = "true"  style = "height:200px;"  bindscrolltoupper = "upper"  bindscrolltolower = " lower"  bindscroll  = " scroll"  scroll  -  into  -  view  = "  {{ toView }}"  scroll  -  top  = "
{{scrollTop}}" >
<viewid = "green"  class = "scroll - view - item bc_green" > </view >
<viewid = "red"  class = "scroll - view - item bc_red" > </view >
<viewid = "yellow"  class = "scroll - view - item bc_yellow" > </view >
<viewid = "blue"  class = "scroll - view - item bc_blue" > </view >
</scroll - view >
```

```
<viewclass = "btn - area" >
<buttonsize = "mini" bindtap = "tap" >click me to scroll into view </button >
<buttonsize = "mini" bindtap = "tapMove" >click me to scroll </button >
</view >
</view >
<viewclass = "section section_gap" >
<viewclass = "section__title" >horizontal scroll </view >
<scroll - viewclass = "scroll - view_H" scroll - x = "true" style = "width:100%" >
<viewid = "green" class = "scroll - view - item_H bc_green" > </view >
<viewid = "red" class = "scroll - view - item_H bc_red" > </view >
<viewid = "yellow" class = "scroll - view - item_H bc_yellow" > </view >
<viewid = "blue" class = "scroll - view - item_H bc_blue" > </view >
</scroll - view >
</view >
var order = ['red','yellow','blue','green','red']
Page({
  data:{
    toView:'red',
    scrollTop:100
  },
  upper:function(e){
console.log(e)
  },
  lower:function(e){
console.log(e)
  },
  scroll:function(e){
console.log(e)
  },
  tap:function(e){
for(var i=0;i<order.length;++i){
if(order[i]===this.data.toView){
this.setData({
        toView:order[i+1]
      })
break
    }
  }
  },
  tapMove:function(e){
this.setData({
    scrollTop:this.data.scrollTop+10
```

```
    })
  }
})
```

3. Bug & Tip

(1) Tip：请勿在 scroll - view 中使用 textarea、map、canvas、video 组件。

(2) Tip：scroll - into - view 的优先级高于 scroll - top。

A.1.3　swiper

1. swiper

滑块视图容器组件如表 A-3 所示。

表 A-3　滑块视图容器组件

属 性 名	类　型	默 认 值	说　明
indicator - dots	Boolean	false	是否显示面板指示点
autoplay	Boolean	false	是否自动切换
current	Number	0	当前所在页面的 index
interval	Number	5000	自动切换时间间隔
duration	Number	1000	滑动动画时长
bindchange	EventHandle		current 改变时会触发 change 事件，event. detail = {current:current}

注意：其中只可放置 <swiper - item/ > 组件，其他节点会被自动删除。

2. swiper - item

仅可放置在 <swiper/ > 组件中，宽高自动设置为 100%。

3. 示例代码 A-3

```
<swiperindicator - dots = "{{indicatorDots}}"
autoplay = "{{autoplay}}" interval = "{{interval}}" duration = "{{duration}}" >
<block wx:for = "{{imgUrls}}" >
<swiper - item >
<imagesrc = "{{item}}" class = "slide - image" width = "355" height = "150"/ >
</swiper - item >
</block >
</swiper >
<buttonbindtap = "changeIndicatorDots" > indicator - dots </button >
<buttonbindtap = "changeAutoplay" > autoplay </button >
<sliderbindchange = "intervalChange" show - value min = "500" max = "2000"/ > interval
<sliderbindchange = "durationChange" show - value min = "1000" max = "10000"/ > duration
Page({
```

```
  data:{
    imgUrls:[
'http://img02. tooopen. com/images/20150928/tooopen_sy_143912755726. jpg',
'http://img06. tooopen. com/images/20160818/tooopen_sy_175866434296. jpg',
'http://img06. tooopen. com/images/20160818/tooopen_sy_175833047715. jpg'
    ],
    indicatorDots:false,
    autoplay:false,
    interval:5000,
    duration:1000
  },
  changeIndicatorDots:function(e){
this. setData({
      indicatorDots:!this. data. indicatorDots
    })
  },
  changeAutoplay:function(e){
this. setData({
      autoplay:!this. data. autoplay
    })
  },
  intervalChange:function(e){
this. setData({
      interval:e. detail. value
    })
  },
  durationChange:function(e){
this. setData({
      duration:e. detail. value
    })
  }
})
```

A. 2 基础内容组件

A. 2. 1 icon

1. icon

图标组件如表 A-4 所示。



表 A-4　图标组件

属 性 名	类　型	默 认 值	说　明
type	String		icon 的类型，有效值：success、success_no_circle、info、warn、waiting、cancel、download、search、clear
size	Number	23	icon 的大小，单位 px
color	Color		icon 的颜色，同 css 的 color

2. 示例代码 A-4

```
<viewclass = "group" >
<blockwx:for = "{{iconSize}}" >
<icontype = "success" size = "{{item}}"/>
</block >
</view >

<view class = "group" >
<blockwx:for = "{{iconType}}" >
<icontype = "{{item}}" size = "45"/>
</block >
</view >

<viewclass = "group" >
<blockwx:for = "{{iconColor}}" >
<icontype = "success" size = "45" color = "{{item}}"/>
</block >
</view >
Page({
  data:{
    iconSize:[20,30,40,50,60,70],
    iconColor:[
'red','orange','yellow','green','rgb(0,255,255)','blue','purple'
    ],
    iconType:[
'success','info','warn','waiting','safe_success',
'safe_warn','success_circle','success_no_circle','waiting_circle',
'circle','download',
'info_circle','cancel','search','clear'
    ]
  }
})
```

A.2.2　text

1. text

文本组件，支持转义符“\”，<text/>组件内只支持 <text/>嵌套。除了文本节点以外的其他节点都无法长按选中。

2. 示例代码 A-5

```
<viewclass = "btn - area" >
<viewclass = "body - view" >
<text > {{text}} </text >
<buttonbindtap = "add" > add line </button >
<buttonbindtap = "remove" > remove line </button >
</view >
</view >
var initData = 'this is first line\nthis is second line '
var extraLine = [ ];
Page( {
  data: {
    text: initData
  },
  add: function( e) {
    extraLine. push('other line ')
this. setData( {
      text: initData + '\n ' + extraLine. join('\n ')
    })
  },
  remove: function( e) {
if( extraLine. length > 0) {
      extraLine. pop( )
this. setData( {
        text: initData + '\n ' + extraLine. join('\n ')
      })
    }
  }
})
```

A.2.3　progress

1. progress

进度条组件如表 A-5 所示。

表 A-5　进度条组件

属 性 名	类　型	默 认 值	说　明
percent	Float	无	百分比 0 ~ 100
show - info	Boolean	false	在进度条右侧显示百分比
stroke - width	Number	6	进度条线的宽度，单位 px
color	Color	#09BB07	进度条颜色
active	Boolean	false	进度条从左往右的动画

2. 示例代码 A-6

```
<progresspercent = "20" show - info / >
<progresspercent = "40" stroke - width = "12" / >
<progresspercent = "60" color = "pink" / >
<progresspercent = "80" active / >
```

A.3　表单组件

A.3.1　button

1. button

按钮组件如表 A-6 所示。

表 A-6　按钮组件

属 性 名	类　型	默 认 值	说　明
size	String	default	有效值 default、mini
type	String	default	按钮的样式类型，有效值 primary、default、warn
plain	Boolean	false	按钮是否镂空，背景色透明
disabled	Boolean	false	是否禁用
loading	Boolean	false	名称前是否带 loading 图标
form - type	String	无	有效值：submit、reset，用于 < form/ > 组件，单击分别会触发 submit/reset 事件
hover - class	String	button - hover	指定按钮按下去的样式类。当 hover - class = "none" 时，没有单击态效果

注意： button - hover 默认为{background - color:rgba(0,0,0,0.1);opacity:0.7;}

2. 示例代码 A-7

```
/ * * wxss * * /
/ * * 修改 button 默认的单击态样式类 * * /
. button - hover{
```

```
background - color:red;
}
/ * * 添加自定义 button 单击态样式类 * * /
. other - button - hover{
background - color:blue;
}
<button type = "default" size = "{{defaultSize}}" loading = "{{loading}}" plain = "{{plain}}"
disabled = "{{disabled}}" bindtap = "default" hover - class = "other - button - hover" > default
</button >
<buttontype = "primary" size = "{{primarySize}}" loading = "{{loading}}" plain = "{{plain}}"
disabled = "{{disabled}}" bindtap = "primary" > primary </button >
<buttontype = "warn" size = "{{warnSize}}" loading = "{{loading}}" plain = "{{plain}}"
disabled = "{{disabled}}" bindtap = "warn" > warn </button >
<buttonbindtap = "setDisabled" > 单击设置以上按钮 disabled 属性 </button >
<buttonbindtap = "setPlain" > 单击设置以上按钮 plain 属性 </button >
<buttonbindtap = "setLoading" > 单击设置以上按钮 loading 属性 </button >
var types = ['default','primary','warn']
var pageObject = {
  data:{
    defaultSize:'default',
    primarySize:'default',
    warnSize:'default',
    disabled:false,
    plain:false,
    loading:false
  },
  setDisabled:function(e){
this.setData({
      disabled:!this.data.disabled
    })
  },
  setPlain:function(e){
this.setData({
      plain:!this.data.plain
    })
  },
  setLoading:function(e){
this.setData({
      loading:!this.data.loading
    })
  }
}
```

```
for( var i =0;i < types. length; ++ i) {
   (function( type) {
      pageObject[ type] = function( e) {
var key = type + 'Size '
var changedData = {}
         changedData[ key] =
this. data[ key] === 'default '? 'mini ':'default '
this. setData( changedData)
      }
   }) (types[ i])
}

Page( pageObject)
```

A. 3. 2　checkbox – group

1. checkbox – group

多项选择器，内部由多个 checkbox 组成，如表 A-7 所示。

表 A-7　多项选择器

属 性 名	类　型	默 认 值	说　明
bindchange	EventHandle		< checkbox – group/ > 中选中项发生改变时触发 change 事件，detail = {value:[选中的 checkbox 的 value 的数组]}

2. checkbox

多选项目如表 A-8 所示。

表 A-8　checkbox 多选项目

属 性 名	类　型	默 认 值	说　明
value	String		< checkbox/ > 标识，选中时触发 < checkbox – group/ > 的 change 事件，并携带 < checkbox/ > 的 value
disabled	Boolean	false	是否禁用
checked	Boolean	false	当前是否选中，可用来设置默认选中

3. 示例代码 A-8

```
< checkbox – groupbindchange = " checkboxChange" >
< labelclass = " checkbox"  wx:for = " {{items}} " >
< checkboxvalue = " {{item. name}} "  checked = " {{item. checked}} " / > {{item. value}}
</label >
```

```
</checkbox - group >
Page({
  data:{
    items:[
      {name:'USA',value:'美国'},
      {name:'CHN',value:'中国',checked:'true'},
      {name:'BRA',value:'巴西'},
      {name:'JPN',value:'日本'},
      {name:'ENG',value:'英国'},
      {name:'TUR',value:'法国'},
    ]
  },
  checkboxChange:function(e){
console.log('checkbox 发生 change 事件,携带 value 值为:',e.detail.value)
  }
})
```

A.3.3 form

1. form

表单组件。将组件内的用户输入的 <switch/> <input/> <checkbox/> <slider/> <radio/> <picker/> 提交。

当单击 <form/> 表单中 formType 为 submit 的 <button/> 组件时，会将表单组件中的 value 值进行提交，需要在表单组件中加上 name 作为 key。如表 A-9 所示。

表 A-9 form 表单组件

属 性 名	类 型	说 明
report - submit	Boolean	是否返回 formId 用于发送模板消息
bindsubmit	EventHandle	携带 form 中的数据触发 submit 事件，event.detail = {value:{'name':'value'},formId:''}
bindreset	EventHandle	表单重置时会触发 reset 事件

2. 示例代码 A-9

```
<formbindsubmit = "formSubmit" bindreset = "formReset" >
<viewclass = "section section_gap" >
<viewclass = "section__title" >switch </view >
<switchname = "switch"/ >
</view >
<viewclass = "section section_gap" >
```

```
<view class = "section__title" > slider </view >
<slider name = "slider" show - value > </slider >
</view >

<view class = "section" >
<view class = "section__title" > input </view >
<input name = "input" placeholder = "please input here" / >
</view >
<view class = "section section_gap" >
<view class = "section__title" > radio </view >
<radio - group name = "radio - group" >
<label > <radio value = "radio1" / > radio1 </label >
<label > <radio value = "radio2" / > radio2 </label >
</radio - group >
</view >
<view class = "section section_gap" >
<view class = "section__title" > checkbox </view >
<checkbox - group name = "checkbox" >
<label > <checkbox value = "checkbox1" / > checkbox1 </label >
<label > <checkbox value = "checkbox2" / > checkbox2 </label >
</checkbox - group >
</view >
<view class = "btn - area" >
<button formType = "submit" > Submit </button >
<button formType = "reset" > Reset </button >
</view >
</form >
Page({
  formSubmit: function(e) {
console.log('form 发生了 submit 事件,携带数据为:',e.detail.value)
  },
  formReset: function() {
console.log('form 发生了 reset 事件')
  }
})
```

A.3.4　input

1. input

输入框组件，如表 A-10 所示。

表 A-10　输入框组件

属　性　名	类　　型	默　认　值	说　　明
value	String		输入框的初始内容
type	String	text	input 的类型，有效值：text，number，idcard，digit
password	Boolean	false	是否是密码类型
placeholder	String		输入框为空时占位符
placeholder – style	String		指定 placeholder 的样式
placeholder – class	String	input – placeholder	指定 placeholder 的样式类
disabled	Boolean	false	是否禁用
maxlength	Number	140	最大输入长度，设置为 -1 的时候不限制最大长度
auto – focus	Boolean	false	自动聚焦，拉起键盘。页面中只能有一个 < input/ > 或 < textarea/ > 设置 auto – focus 属性
focus	Boolean	false	获取焦点（开发工具暂不支持）
bindinput	EventHandle		除了 date/time 类型外的输入框，当键盘输入时，触发 input 事件，event. detail = {value:value}，处理函数可以直接 return 一个字符串，将替换输入框的内容。
bindfocus	EventHandle		输入框聚焦时触发，event. detail = {value:value}
bindblur	EventHandle		输入框失去焦点时触发，event. detail = {value:value}

2. 示例代码 A-10

```
<!-- input. wxml -- >
<viewclass = "section" >
<inputplaceholder = "这是一个可以自动聚焦的 input" auto – focus/ >
</view >
<viewclass = "section" >
<inputplaceholder = "这个只有在按钮单击的时候才聚焦" focus = "{{focus}}" / >
<viewclass = "btn – area" >
<buttonbindtap = "bindButtonTap" > 使得输入框获取焦点 </button >
</view >
</view >
<viewclass = "section" >
<inputmaxlength = "10" placeholder = "最大输入长度 10" / >
</view >
<viewclass = "section" >
<viewclass = "section__title" > 你输入的是:{{inputValue}} </view >
<inputbindinput = "bindKeyInput" placeholder = "输入同步到 view 中"/ >
</view >
<viewclass = "section" >
<inputbindinput = "bindReplaceInput" placeholder = "连续的两个 1 会变成 2" / >
</view >
<viewclass = "section" >
<inputbindinput = "bindHideKeyboard" placeholder = "输入 123 自动收起键盘" / >
```

```
</view>
<viewclass="section">
<inputpasswordtype="number" />
</view>
<viewclass="section">
<inputpassword type="text" />
</view>
<viewclass="section">
<inputtype="digit" placeholder="带小数点的数字键盘"/>
</view>
<viewclass="section">
<inputtype="idcard" placeholder="身份证输入键盘"/>
</view>
<viewclass="section">
<inputplaceholder-style="color:red" placeholder="占位符字体是红色的" />
</view>
//input.js
Page({
  data:{
    focus:false,
    inputValue:''
  },
  bindButtonTap:function(){
this.setData({
      focus:true
    })
  },
  bindKeyInput:function(e){
this.setData({
      inputValue:e.detail.value
    })
  },
  bindReplaceInput:function(e){
var value=e.detail.value
var pos=e.detail.cursor
if(pos!=-1){
//光标在中间
var left=e.detail.value.slice(0,pos)
//计算光标的位置
      pos=left.replace(/11/g,'2').length
    }
```

```
    //直接返回对象,可以对输入进行过滤处理,同时可以控制光标的位置
    return {
            value:value. replace(/11/g,'2'),
            cursor:pos
        }

    //或者直接返回字符串,光标在最后边
    //return value. replace(/11/g,'2'),
      },
      bindHideKeyboard:function(e) {
    if (e. detail. value  === '123') {
    //收起键盘
            wx. hideKeyboard()
        }
      }
    })
```

3. Bug & Tip

（1）Bug：微信版本 6.3.30，focus 属性设置无效；

（2）Bug：微信版本 6.3.30，placeholder 在聚焦时出现重影问题；

（3）Tip：input 组件是一个 native 组件，字体是系统字体，所以无法设置 font - family；

（4）Tip：在 input 聚焦期间，避免使用 css 动画。

A.3.5　label

1. label

标签组件。

用来改进表单组件的可用性，使用 for 属性找到对应的 id，或者将控件放在该标签下，当单击时，就会触发对应的控件。如表 A-11 所示。

for 优先级高于内部控件，内部有多个控件的时候默认触发第一个控件。

目前可以绑定的控件有：< button/>，< checkbox/>，< radio/>，< switch/>。

表 A-11　label 标签组件

属　性　名	类　　型	说　　明
for	String	绑定控件的 id

2. 示例代码 A-11

```
    < viewclass = " section section_gap" >
```

```
<viewclass = "section__title" >表单组件在 label 内 </view >
<checkbox - group class = "group"  bindchange = "checkboxChange" >
<viewclass = "label - 1"  wx:for = "{{checkboxItems}}" >
<label >
<checkboxhiddenvalue = "{{item. name}}"  checked = "{{item. checked}}" > </checkbox >
<viewclass = "label - 1__icon" >
<viewclass = "label - 1__icon - checked"  style = "opacity:{{item. checked? 1:0}}" > </view >
</view >
<text class = "label - 1__text" >{{item. value}} </text >
</label >
</view >
</checkbox - group >
</view >

<viewclass = "section section_gap" >
<viewclass = "section__title" >label 用 for 标识表单组件 </view >
<radio - groupclass = "group"  bindchange = "radioChange" >
<viewclass = "label - 2"  wx:for = "{{radioItems}}" >
<radioid = "{{item. name}}"  hiddenvalue = "{{item. name}}"  checked = "{{item. checked}}" >
</radio >
<viewclass = "label - 2__icon" >
<viewclass = "label - 2__icon - checked"  style = "opacity:{{item. checked? 1:0}}" > </view >
</view >
<labelclass = "label - 2__text"  for = "{{item. name}}" > <text >{{item. name}} </text > </label >
</view >
</radio - group >
</view >
Page({
  data:{
    checkboxItems:[
      {name:'USA',value:'美国'},
      {name:'CHN',value:'中国',checked:'true'},
      {name:'BRA',value:'巴西'},
      {name:'JPN',value:'日本',checked:'true'},
      {name:'ENG',value:'英国'},
      {name:'TUR',value:'法国'},
    ],
    radioItems:[
      {name:'USA',value:'美国'},
      {name:'CHN',value:'中国',checked:'true'},
      {name:'BRA',value:'巴西'},
      {name:'JPN',value:'日本'},
```

```
        {name:'ENG',value:'英国'},
        {name:'TUR',value:'法国'},
      ],
      hidden:false
    },
    checkboxChange:function(e) {
var checked = e.detail.value
var changed = {}
for (var i =0;i < this.data.checkboxItems.length;i ++) {
if (checked.indexOf(this.data.checkboxItems[i].name) !== -1) {
          changed['checkboxItems[' + i +'].checked'] = true
        }else {
          changed['checkboxItems[' + i +'].checked'] = false
        }
      }
this.setData(changed)
    },
    radioChange:function(e) {
var checked = e.detail.value
var changed = {}
for (var i =0;i < this.data.radioItems.length;i ++) {
if (checked.indexOf(this.data.radioItems[i].name) !== -1) {
          changed['radioItems[' + i +'].checked'] = true
        }else {
          changed['radioItems[' + i +'].checked'] = false
        }
      }
this.setData(changed)
    }
})
.label-1,.label-2{
margin-bottom:15px;
}
.label-1__text,.label-2__text {
display:inline-block;
vertical-align:middle;
}

.label-1__icon {
position:relative;
margin-right:10px;
display:inline-block;
```

```
vertical - align:middle;
width:18px;
height:18px;
background:#fcfff4;
}

.label - 1__icon - checked {
position:absolute;
top:3px;
left:3px;
width:12px;
height:12px;
background:#1aad19;
}


.label - 2__icon {
position:relative;
display:inline - block;
vertical - align:middle;
margin - right:10px;
width:18px;
height:18px;
background:#fcfff4;
border - radius:50px;
}

.label - 2__icon - checked {
position:absolute;
left:3px;
top:3px;
width:12px;
height:12px;
background:#1aad19;
border - radius:50%;
}

.label - 4_text{
text - align:center;
margin - top:15px;
}
```

A.3.6　picker

1. picker

滚动选择器组件。

现支持三种选择器，通过 mode 来区分，分别是普通选择器、时间选择器、日期选择器，默认是普通选择器。

（1）普通选择器：mode = selector，如表 A-12 所示。

表 A-12　普通选择器

属 性 名	类　型	默认值	说　明
range	Array	[]	mode 为 selector 时，range 有效
value	Number	0	mode 为 selector 时，是数字，表示选择了 range 中的第几个，从 0 开始
bindchange	EventHandle		value 改变时触发 change 事件，event. detail = {value：value}

（2）时间选择器：mode = time，如表 A-13 所示。

表 A-13　时间选择器

属 性 名	类　型	默认值	说　明
value	String		表示选中的时间，格式为"hh:mm"
start	String		表示有效时间范围的开始，字符串格式为"hh:mm"
end	String		表示有效时间范围的结束，字符串格式为"hh:mm"
bindchange	EventHandle		value 改变时触发 change 事件，event. detail = {value:value}

（3）日期选择器：mode = date，如表 A-14 所示。

表 A-14　A 期选择器

属 性 名	类　型	默认值	说　明
value	String	0	表示选中的日期，格式为"YYYY - MM - DD"
start	String		表示有效日期范围的开始，字符串格式为"YYYY - MM - DD"
end	String		表示有效日期范围的结束，字符串格式为"YYYY - MM - DD"
fields	String	day	有效值 year，month，day，表示选择器的粒度
bindchange	EventHandle		value 改变时触发 change 事件，event. detail = {value:value}

注意：开发工具暂时只支持 mode = selector。

2. 示例代码 A-12

```
<viewclass = "section" >
<viewclass = "section__title" >地区选择器 </view >
<pickerbindchange = "bindPickerChange" value = "{{index}}" range = "{{array}}" >
<viewclass = "picker" >
```

```
当前选择:{{array[index]}}
</view>
</picker>
</view>
<viewclass="section">
<viewclass="section__title">时间选择器</view>
<pickermode="time" value="{{time}}" start="09:01" end="21:01" bindchange="bindTi-
meChange">
<viewclass="picker">
当前选择:{{time}}
</view>
</picker>
</view>

<viewclass="section">
<viewclass="section__title">日期选择器</view>
<pickermode="date" value="{{date}}" start="2015-09-01" end="2017-09-01" bind-
change="bindDateChange">
<view class="picker">
当前选择:{{date}}
</view>
</picker>
</view>
Page({
  data:{
    array:['美国','中国','巴西','日本'],
    index:0,
    date:'2016-09-01',
    time:'12:01'
  },
  bindPickerChange:function(e){
console.log('picker 发送选择改变,携带值为',e.detail.value)
this.setData({
      index:e.detail.value
    })
  },
  bindDateChange:function(e){
this.setData({
      date:e.detail.value
    })
  },
  bindTimeChange:function(e){
```

```
    this. setData( {
        time:e. detail. value
      } )
    }
} )
```

A. 3. 7　radio – group

1. radio – group

单项选择器组件，内部由多个 < radio/ > 组成。如表 A–15 所示。

表 A–15　单项选择器组件

属　性　名	类　　型	默认值	说　　明
bindchange	EventHandle		< radio – group/ > 中的选中项发生变化时触发 change 事件，event. detail = {value:选中项 radio 的 value}

2. radio

单选项目组件，如表 A–16 所示。

表 A–16　单选项目组件

属　性　名	类　　型	默认值	说　　明
value	String		< radio/ > 标识。当该 < radio/ > 选中时，< radio – group/ > 的 change 事件会携带 < radio/ > 的 value
checked	Boolean	false	当前是否选中
disabled	Boolean	false	是否禁用

3. 示例代码 A–13

```
< radio – group class = " radio – group"  bindchange = " radioChange" >
< labelclass = " radio"  wx:for = " {{items}} " >
< radiovalue = " {{item. name}} "  checked = " {{item. checked}} " /> {{item. value}}
</label >
</radio – group >
Page( {
  data:{
    items:[
      {name:'USA',value:'美国'},
      {name:'CHN',value:'中国',checked:'true'},
      {name:'BRA',value:'巴西'},
      {name:'JPN',value:'日本'},
```

```
        {name:'ENG',value:'英国'},
        {name:'TUR',value:'法国'},
      ]
    },
    radioChange:function(e) {
console.log('radio 发生 change 事件,携带 value 值为:',e.detail.value)
    }
})
```

A.3.8　slider

1. slider

滑动选择器组件，如表 A-17 所示。

表 A-17　滑动选择器组件

属　性　名	类　　型	默认值	说　　明
min	Number	0	最小值
max	Number	100	最大值
step	Number	1	步长，取值必须大于0，并且可被（max - min）整除
disabled	Boolean	false	是否禁用
value	Number	0	当前取值
show - value	Boolean	false	是否显示当前 value
bindchange	EventHandle		完成一次拖动后触发的事件，event.detail = {value:value}

2. 示例代码 A-14

```
<viewclass = "section section_gap" >
<textclass = "section__title" >设置 left/right icon </text >
<viewclass = "body - view" >
<sliderbindchange = "slider1change" left - icon = "cancel" right - icon = "success_no_circle"/>
</view >
</view >

<viewclass = "section section_gap" >
<textclass = "section__title" >设置 step </text >
<viewclass = "body - view" >
<sliderbindchange = "slider2change" step = "5"/>
</view >
</view >
```

```
<viewclass = "section section_gap" >
<textclass = "section__title" > 显示当前 value </text >
<viewclass = "body - view" >
<sliderbindchange = "slider3change" show - value/>
</view >
</view >

<viewclass = "section section_gap" >
<textclass = "section__title" > 设置最小/最大值 </text >
<viewclass = "body - view" >
<sliderbindchange = "slider4change" min = "50" max = "200" show - value/>
</view >
</view >
var pageData = { }
for (var i = 1;i < 5;i ++) {
  (function (index) {
    pageData['slider' + index + 'change'] = function(e) {
console.log('slider' + 'index' + '发生 change 事件,携带值为',e.detail.value)
    }
  })(i)
}
Page(pageData)
```

A.3.9　switch

1. switch

开关选择器组件，如表 A-18 所示。

表 A-18　开关选择器组件

属　性　名	类　　型	默认值	说　　明
checked	Boolean	false	是否选中
type	String	switch	样式，有效值：switch、checkbox
bindchange	EventHandle		checked 改变时触发 change 事件，event.detail = {value:checked}

2. 示例代码 A-15

```
<viewclass = "body - view" >
<switchcheckedbindchange = "switch1Change"/>
<switchbindchange = "switch2Change"/>
```

```
</view>
Page({
  switch1Change:function (e){
console.log('switch1 发生 change 事件,携带值为',e.detail.value)
  },
  switch2Change:function (e){
console.log('switch2 发生 change 事件,携带值为',e.detail.value)
  }
})
```

A.3.10　textarea

1. textarea

多行输入框组件，如表 A-19 所示。

表 A-19　多行输入框组件

属 性 名	类　型	默 认 值	说　明
value	String		输入框的内容
placeholder	String		输入框为空时占位符
placeholder-style	String		指定 placeholder 的样式
placeholder-class	String	textarea-placeholder	指定 placeholder 的样式类
disabled	Boolean	false	是否禁用
maxlength	Number	140	最大输入长度，设置为 0 的时候不限制最大长度
auto-focus	Boolean	false	自动聚焦，拉起键盘。页面中只能有一个 <textarea/> 或 <input/> 设置 auto-focus 属性
focus	Boolean	false	获取焦点（开发工具暂不支持）
auto-height	Boolean	false	是否自动增高，设置 auto-height 时，style.height 不生效
bindfocus	EventHandle		输入框聚焦时触发，event.detail = {value:value}
bindblur	EventHandle		输入框失去焦点时触发，event.detail = {value:value}
bindlinechange	EventHandle		输入框行数变化时调用，event.detail = {height:0,heightRpx:0,lineCount:0}

2. 示例代码 A-16

```
<!--textarea.wxml-->
<viewclass="section">
<textareabindblur="bindTextAreaBlur" auto-heightplaceholder="自动变高" />
</view>
```

```
<viewclass = "section" >
<textareaplaceholder = "placeholder 颜色是红色的" placeholder - style = "color:red;"  />
</view >
<viewclass = "section" >
<textareaplaceholder = "这是一个可以自动聚焦的 textarea" auto - focus/>
</view >
<viewclass = "section" >
<textareaplaceholder = "这个只有在按钮单击的时候才聚焦" focus = "{{focus}}" />
<viewclass = "btn - area" >
<buttonbindtap = "bindButtonTap" >使得输入框获取焦点 </button >
</view >
</view >
//textarea. js
Page({
  data:{
    height:20,
    focus:false
  },
  bindButtonTap:function() {
this. setData({
      focus:true
    })
  },
  bindTextAreaBlur:function(e) {
console. log(e. detail. value)
  }
})
```

3. Bug & Tip

（1）Bug：微信版本 6.3.30，textarea 在列表渲染时，新增加的 textarea 在自动聚焦时的位置计算错误；

（2）Tip：请勿在 scroll - view 中使用 textarea 组件。

A.4　导航组件

1. navigator

页面链接组件，如表 A-20 所示。



表 A-20　页面链接组件

属 性 名	类　型	默 认 值	说　明
url	String		应用内的跳转链接
redirect	Boolean	false	是否关闭当前页面
hover - class	String	navigator - hover	指定单击时的样式类，当 hover - class = "none" 时，没有单击态效果

注： navigator - hover 默认为{background - color:rgba(0,0,0,0.1);opacity:0.7;}，<navigator/>的子节点背景色应为透明色。

2. 示例代码 A-17

```
/** wxss **/
/** 修改默认的 navigator 单击态 **/
.navigator - hover {
color:blue;
}
/** 自定义其他单击态样式类 **/
.other - navigator - hover {
color:red;
}
<!-- sample.wxml -->
<viewclass = "btn - area" >
<navigatorurl = "navigate? title = navigate" hover - class = "navigator - hover" >跳转到新页面
</navigator >
<navigatorurl = "redirect? title = redirect" redirecthover - class = "other - navigator - hover" >在当前页打开 </navigator >
</view >
<!-- navigator.wxml -->
<viewstyle = "text - align:center" > {{title}} </view >
<view > 单击左上角返回到之前页面 </view >
<!-- redirect.wxml -->
<viewstyle = "text - align:center" > {{title}} </view >
<view > 单击左上角返回到上级页面 </view >
// redirect.js navigator.js
Page({
  onLoad:function(options) {
this.setData({
      title:options.title
    })
  }
})
```

A.5　媒体组件

A.5.1　image

1. image

图片组件，如表 A-21 所示。

表 A-21　图片组件

属 性 名	类　型	默 认 值	说　明
src	String		图片资源地址
mode	String	'scaleToFill'	图片裁剪、缩放的模式
binderror	HandleEvent		当错误发生时，发布到 AppService 的事件名，事件对象 event. detail = {errMsg:'something wrong'}
bindload	HandleEvent		当图片载入完毕时，发布到 AppService 的事件名，事件对象 event. detail = {height:'图片高度 px',width:'图片宽度 px'}

注： image 组件默认宽度 300px、高度 225px。

mode 有 12 种模式，其中 3 种是缩放模式，9 种是裁剪模式，如表 A-22 和表 A-23 所示。

表 A-22　缩放模式

模　式	说　明
scaleToFill	不保持纵横比缩放图片，使图片的宽高完全拉伸至填满 image 元素
aspectFit	保持纵横比缩放图片，使图片的长边能完全显示出来。也就是说，可以完整地将图片显示出来
aspectFill	保持纵横比缩放图片，只保证图片的短边能完全显示出来。也就是说，图片通常只在水平或垂直方向是完整的，另一个方向将会发生截取

表 A-23　裁剪模式

模　式	说　明
top	不缩放图片，只显示图片的顶部区域
bottom	不缩放图片，只显示图片的底部区域
center	不缩放图片，只显示图片的中间区域
left	不缩放图片，只显示图片的左边区域
right	不缩放图片，只显示图片的右边区域
top left	不缩放图片，只显示图片的左上边区域
top right	不缩放图片，只显示图片的右上边区域
bottom left	不缩放图片，只显示图片的左下边区域
bottom right	不缩放图片，只显示图片的右下边区域

2. 示例代码 A-18

```
<viewclass = "page" >
<viewclass = "page__hd" >
<textclass = "page__title" > image </text >
<textclass = "page__desc" > 图片 </text >
</view >
<viewclass = "page__bd" >
<viewclass = "section section_gap" wx:for = "{{array}}" wx:for - item = "item" >
<viewclass = "section__title" > {{item. text}} </view >
<viewclass = "section__ctn" >
<imagestyle = "width:200px;height:200px;background - color:#eeeeee;" mode = "{{item. mode}}"
src = "{{src}}" > </image >
</view >
</view >
</view >
</view >
Page({
  data:{
    array:[{
      mode:'scaleToFill',
      text:'scaleToFill:不保持纵横比缩放图片,使图片完全适应'
    },{
      mode:'aspectFit',
      text:'aspectFit:保持纵横比缩放图片,使图片的长边能完全显示出来'
    },{
      mode:'aspectFill',
      text:'aspectFill:保持纵横比缩放图片,只保证图片的短边能完全显示出来'
    },{
      mode:'top',
      text:'top:不缩放图片,只显示图片的顶部区域'
    },{
      mode:'bottom',
      text:'bottom:不缩放图片,只显示图片的底部区域'
    },{
      mode:'center',
      text:'center:不缩放图片,只显示图片的中间区域'
    },{
      mode:'left',
      text:'left:不缩放图片,只显示图片的左边区域'
    },{
```

```
        mode:'right',
        text:'right:不缩放图片,只显示图片的右边边区域'
      },{
        mode:'top left',
        text:'top left:不缩放图片,只显示图片的左上边区域'
      },{
        mode:'top right',
        text:'top right:不缩放图片,只显示图片的右上边区域'
      },{
        mode:'bottom left',
        text:'bottom left:不缩放图片,只显示图片的左下边区域'
      },{
        mode:'bottom right',
        text:'bottom right:不缩放图片,只显示图片的右下边区域'
      }],
      src:'../../resources/cat.jpg'
    },
    imageError:function(e) {
console.log('image3 发生 error 事件,携带值为',e.detail.errMsg)
    }
})
```

A.5.2　video

1. video

视频组件，如表 A-24 所示。

表 A-24　视频组件

属　性　名	类　　型	默认值	说　　明
src	String		要播放视频的资源地址
controls	Boolean	true	是否显示默认播放控件（播放/暂停按钮、播放进度、时间）
danmu - list	Object Array		弹幕列表
danmu - btn	Boolean	false	是否显示弹幕按钮，只在初始化时有效，不能动态变更
enable - danmu	Boolean	false	是否展示弹幕，只在初始化时有效，不能动态变更
autoplay	Boolean	false	是否自动播放
bindplay	EventHandle		当开始/继续播放时触发 play 事件
bindpause	EventHandle		当暂停播放时触发 pause 事件
bindended	EventHandle		当播放到末尾时触发 ended 事件
binderror	EventHandle		当发生错误时触发 error 事件，event.detail = {errMsg:'something wrong'}

video 标签默认宽度 300px、高度 225px，设置宽高需要通过 wxss 设置 width 和 height。

2. 示例代码 A-19

```
<viewclass = "section tc" >
<videosrc = "{{src}}" controls > </video >
<view class = "btn - area" >
<buttonbindtap = "bindButtonTap" > 获取视频 </button >
</view >
</view >

<viewclass = "section tc" >
<videoid = "myVideo" src = " http://wxsnsdy. tc. qq. com/105/20210/snsdyvideodownload? filekey =
30280201010421301f0201690402534804102ca905ce620b1241b726bc41dcff44e00204012882540400&bizid
= 1023&hy = SH&fileparam = 302c020101042530230204136ffd93020457e3c4ff02024ef202031e8d7f02030
f42400204045a320a0201000400" binderror = "videoErrorCallback" danmu - list = "{{danmuList}}" en-
able - danmu danmu - btn controls > </video >
<viewclass = "btn - area" >
<buttonbindtap = "bindButtonTap" > 获取视频 </button >
<inputbindblur = "bindInputBlur"/>
<buttonbindtap = "bindSendDanmu" > 发送弹幕 </button >
</view >
</view >
functiongetRandomColor ( ) {
let rgb = [ ]
for (let i = 0 ;i < 3; ++ i) {
let color = Math. floor( Math. random( ) * 256). toString(16)
    color = color. length == 1 ? '0' + color :color
    rgb. push( color)
  }
return '#' + rgb. join('')
}

Page( {
  onReady:function (res) {
this. videoContext = wx. createVideoContext('myVideo')
  },
  inputValue:'',
    data:{
        src:'',
    danmuList:[
      {
```

```
            text:'第 1s 出现的弹幕',
            color:'#ff0000',
            time:1
          },
          {
            text:'第 3s 出现的弹幕',
            color:'#ff00ff',
            time:3
          }
        ]
      },
    bindInputBlur:function(e) {
this.inputValue = e.detail.value
    },
      bindButtonTap:function() {
var that = this
          wx.chooseVideo({
              sourceType:['album','camera'],
              maxDuration:60,
              camera:['front','back'],
              success:function(res) {
                  that.setData({
                      src:res.tempFilePath
                  })
              }
          })
      },
    bindSendDanmu:function () {
this.videoContext.sendDanmu({
        text:this.inputValue,
        color:getRandomColor()
      })
    },
      videoErrorCallback:function(e) {
console.log('视频错误信息:')
console.log(e.detail.errMsg)
      }
})
```

3. Bug & Tip

Tip：请勿在 scroll - view 中使用 video 组件。



A.6　地图组件

1. map

地图组件，如表 A-25 所示。

表 A-25　地图组件

属 性 名	类　型	默 认 值	说　明
longitude	Number		中心经度
latitude	Number		中心纬度
scale	Number	16	缩放级别
markers	Array		标记点
covers	Array		覆盖物

1）标记点

标记点用于在地图上显示标记的位置，不能自定义图标和样式，如表 A-26 所示。

表 A-26　标记点

属　性	说　明	类　型	必　填	备　注
latitude	纬度	Number	是	浮点数，范围 -90 ~ 90
longitude	经度	Number	是	浮点数，范围 -180 ~ 180
name	标注点名	String	是	
desc	标注点详细描述	String	否	

2）覆盖物

覆盖物用于在地图上显示自定义图标，可自定义图标和样式，如表 A-27。

表 A-27　覆盖物

属　性	说　明	类　型	必　填	备　注
latitude	纬度	Number	是	浮点数，范围 -90 ~ 90
longitude	经度	Number	是	浮点数，范围 -180 ~ 180
iconPath	显示的图标	String	是	项目目录下的图片路径，支持相对路径写法
rotate	旋转角度	Number	否	顺时针旋转的角度，范围 0 ~ 360，默认为 0

注：地图组件的经纬度必填，如果不填经纬度则默认值是北京的经纬度。标记点 markers 只能在初始化的时候设置，不支持动态更新。

2. 示例代码 A-20

```
<!-- map.wxml -->
<maplongitude = "113.324520" latitude = "23.099994" markers = "{{markers}}" covers =
```

```
"{{covers}}" style = "width:375px;height:200px;" > </map >
// map.js
Page({
  data:{
    markers:[{
      latitude:23.099994,
      longitude:113.324520,
      name:'T.I.T 创意园',
      desc:'我现在的位置'
    }],
    covers:[{
      latitude:23.099794,
      longitude:113.324520,
      iconPath:'../images/car.png',
      rotate:10
    },{
      latitude:23.099298,
      longitude:113.324129,
      iconPath:'../images/car.png',
      rotate:90
    }]
  }
})
```

3. Bug & Tip

Tip：请勿在 scroll - view 中使用 map 组件。

A.7　画布组件

1. canvas

画布组件，如表 A-28 所示。

表 A-28　画布组件

属　性　名	类　　型	默认值	说　　明
canvas - id	String		canvas 组件的唯一标识符
disable - scroll	Boolean	false	当在 canvas 中移动时，禁止屏幕滚动及下拉刷新
bindtouchstart	EventHandle		手指触摸动作开始
bindtouchmove	EventHandle		手指触摸后移动

续表

属 性 名	类　型	默认值	说　明
bindtouchend	EventHandle		手指触摸动作结束
bindtouchcancel	EventHandle		手指触摸动作被打断，如来电提醒，弹窗
binderror	EventHandle		当发生错误时触发 error 事件，detail = {errMsg:'something wrong'}

注：

（1）canvas 标签默认宽度 300px、高度 225px；

（2）同一页面中的 canvas - id 不可重复，如果使用一个已经出现过的 canvas - id，该 canvas 标签对应的画布将被隐藏并不再正常工作。

2. 示例代码 A-21

```
<!-- canvas.wxml -->
<canvasstyle="width:300px;height:200px;" canvas-id="firstCanvas"></canvas>
<!-- 当使用绝对定位时,文档流后边的 canvas 的显示层级高于前边的 canvas -->
<canvasstyle="width:400px;height:500px;" canvas-id="secondCanvas"></canvas>
<!-- 因为 canvas-id 与前一个 canvas 重复,该 canvas 不会显示,并会发送一个错误事件到 AppService -->
<canvasstyle="width:400px;height:500px;" canvas-id="secondCanvas" binderror="canvasIdErrorCallback"></canvas>
// canvas.js
Page({
  canvasIdErrorCallback:function (e) {
console.error(e.detail.errMsg)
  },
  onReady:function (e) {

// 使用 wx.createContext 获取绘图上下文 context
var context = wx.createContext()

    context.setStrokeStyle("#00ff00")
    context.setLineWidth(5)
    context.rect(0,0,200,200)
    context.stroke()
    context.setStrokeStyle("#ff0000")
    context.setLineWidth(2)
    context.moveTo(160,100)
    context.arc(100,100,60,0,2 * Math.PI,true)
    context.moveTo(140,100)
    context.arc(100,100,40,0,Math.PI,false)
    context.moveTo(85,80)
```

```
        context.arc(80,80,5,0,2 * Math.PI,true)
        context.moveTo(125,80)
        context.arc(120,80,5,0,2 * Math.PI,true)
        context.stroke()

    // 调用 wx.drawCanvas,通过 canvasId 指定在哪张画布上绘制,通过 actions 指定绘制行为
        wx.drawCanvas({
          canvasId:'firstCanvas',
          actions:context.getActions()// 获取绘图动作数组
        })
    }
})
```

3. Bug & Tip

Tip：请勿在 scroll - view 中使用 canvas 组件。

附录 B

小程序 API 详解

微信小程序框架提供丰富的微信原生 API，可以方便地调用微信提供的能力，如获取用户信息，本地存储，支付功能等。

1. 说明与目录

- wx. on 开头的 API 是监听某个事件发生的 API 接口，接受一个 CALLBACK 函数作为参数。当该事件触发时，会调用 CALLBACK 函数；
- 如未特殊约定，其他 API 接口都接受一个 OBJECT 作为参数；
- OBJECT 中可以指定 success、fail、complete 接收接口调用结果，如表 B-1 表示。

表 B-1　OBJECT 指定接收接调用结果

参 数 名	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2. API 列表

（1）网络 API 列表，如表 B-2 所示。

表 B-2　网络 API 列表

API	说　明
wx. request	发起网络请求
wx. uploadFile	上传文件
wx. downloadFile	下载文件
wx. connectSocket	创建 WebSocket 连接
wx. onSocketOpen	监听 WebSocket 打开
wx. onSocketError	监听 WebSocket 错误
wx. sendSocketMessage	发送 WebSocket 消息
wx. onSocketMessage	接受 WebSocket 消息
wx. closeSocket	关闭 WebSocket 连接
wx. onSocketClose	监听 WebSocket 关闭

（2）媒体 API 列表，如表 B-3 所示。

表 B-3　媒体 API 列表

API	说　明
wx. chooseImage	从相册选择图片，或者拍照
wx. previewImage	预览图片
wx. startRecord	开始录音
wx. stopRecord	结束录音
wx. playVoice	播放语音
wx. pauseVoice	暂停播放语音
wx. stopVoice	结束播放语音
wx. getBackgroundAudioPlayerState	获取音乐播放状态
wx. playBackgroundAudio	播放音乐
wx. pauseBackgroundAudio	暂停播放音乐
wx. seekBackgroundAudio	控制音乐播放进度
wx. stopBackgroundAudio	停止播放音乐
wx. onBackgroundAudioPlay	监听音乐开始播放
wx. onBackgroundAudioPause	监听音乐暂停
wx. onBackgroundAudioStop	监听音乐结束
wx. chooseVideo	从相册选择视频，或者拍摄
wx. saveFile	保存文件

（3）数据 API 列表，如表 B-4 所示。

表 B-4　数据 API 列表

API	说　明
wx. getStorage	获取本地数据缓存
wx. setStorage	设置本地数据缓存
wx. clearStorage	清理本地数据缓存

（4）位置 API 列表，如表 B-5 所示。

表 B-5　位置 API 列表

API	说　明
wx. getLocation	获取当前位置
wx. openLocation	打开内置地图

（5）设备 API 列表，如表 B-6 所示。

表 B-6　设备 API 列表

API	说　明
wx. getNetworkType	获取网络类型
wx. getSystemInfo	获取系统信息

续表

API	说　明
wx. onAccelerometerChange	监听重力感应数据
wx. onCompassChange	监听罗盘数据

（6）界面 API 列表，如表 B-7 所示。

表 B-7　界面 API 列表

API	说　明
wx. setNavigationBarTitle	设置当前页面标题
wx. showNavigationBarLoading	显示导航条加载动画
wx. hideNavigationBarLoading	隐藏导航条加载动画
wx. navigateTo	新窗口打开页面
wx. redirectTo	原窗口打开页面
wx. navigateBack	退回上一个页面
wx. createAnimation	动画
wx. createContext	创建绘图上下文
wx. drawCanvas	绘图
wx. hideKeyboard	隐藏键盘
wx. stopPullDownRefresh	停止下拉刷新动画

（7）开放接口，如表 B-8 所示。

表 B-8　开放接口

API	说　明
wx. login	登录
wx. getUserInfo	获取用户信息
wx. requestPayment	发起微信支付

B. 1　网络

每个微信小程序需要事先设置一个通信域名，小程序可以跟指定的域名与网络进行通信。包括普通 HTTPS 请求（wx. request）、WebSocket 通信（wx. connectSocket）、上传文件（wx. uploadFile）和下载文件（wx. downloadFile）。

网络 API 列表如表 B-9 所示。

表 B-9　网络 API 列表

API	说　明
wx. request	发起网络请求

续表

API	说　明
wx. uploadFile	上传文件
wx. downloadFile	下载文件
wx. connectSocket	创建 WebSocket 连接
wx. onSocketOpen	监听 WebSocket 打开
wx. onSocketError	监听 WebSocket 错误
wx. sendSocketMessage	发送 WebSocket 消息
wx. onSocketMessage	接受 WebSocket 消息
wx. closeSocket	关闭 WebSocket 连接
wx. onSocketClose	监听 WebSocket 关闭

B. 1. 1　发起请求

1. wx. request(OBJECT)

wx. request 发起的是 HTTPS 请求。一个微信小程序，同时只能有 5 个网络请求连接。

1）OBJEC 参数说明

OBJECT 参数说明，如表 B-10 所示。

表 B-10　wx. request(OBJECT)中 OBJECT 参数说明

参　数　名	类　型	必填	说　明
url	String	是	开发者服务器接口地址
data	Object、String	否	请求的参数
header	Object	否	设置请求的 header，header 中不能设置 Referer
method	String	否	默认为 GET，有效值：OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT
success	Function	否	收到开发者服务成功返回的回调函数，res = {data:'开发者服务器返回的内容'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-1

```
wx. request({
  url:'test. php',//仅为示例,并非真实的接口地址
  data:{
    x:'',
    y:''
  },
  header:{
```

```
    'content-type':'application/json'
      },
      success:function(res) {
    console.log(res.data)
      }
    })
```

2. Bug & Tip

（1）Bug：开发者工具 0.10.102800 版本，header 的 content-type 设置异常；

（2）Tip：客户端的 HTTPS TLS 版本为 1.2，但 Android 的部分机型还未支持 TLS 1.2，所以请确保 HTTPS 服务器的 TLS 版本支持 1.2 及以下版本；

（3）Tip：要注意 method 的 value 必须为大写（如 GET）；

（4）Tip：url 中不能有端口。

B.1.2　上传、下载

1. wx.uploadFile(OBJECT)

将本地资源上传到开发者服务器。如页面通过 wx.chooseImage 等接口获取到一个本地资源的临时文件路径后，可通过此接口将本地资源上传到指定服务器。客户端发起一个 HTTPS POST 请求，其中 content-type 为 multipart/form-data。

1）OBJECT 参数说明

OBJECT 参数说明，如表 B-11 所示。

表 B-11　wx.uploadFile(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
url	String	是	开发者服务器 url
filePath	String	是	要上传文件资源的路径
name	String	是	文件对应的 key，开发者在服务器端通过这个 key 可以获取到文件二进制内容
header	Object	否	HTTP 请求 Header，header 中不能设置 Referer
formData	Object	否	HTTP 请求中其他额外的 form data
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-12 所示。

表 B-12　success 返回参数说明

参　数	类　型	说　明
data	String	开发者服务器返回的数据
statusCode	Number	HTTP 状态码

3）示例代码 B-2

```
wx. chooseImage({
  success:function(res) {
var tempFilePaths = res. tempFilePaths
    wx. uploadFile({
      url:'http://example. weixin. qq. com/upload',//仅为示例,非真实的接口地址
      filePath:tempFilePaths[0],
      name:'file',
      formData:{
 'user':'test'
      },
      success:function(res){
var data = res. data
//do something
      }
    })
  }
})
```

2. wx. downloadFile(OBJECT)

下载文件资源到本地。客户端直接发起一个 HTTP GET 请求，返回文件的本地临时路径。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-13 所示。

表 B-13　wx. downloadFile(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
url	String	是	下载资源的 url
header	Object	否	HTTP 请求 Header
success	Function	否	下载成功后以 tempFilePath 的形式传给页面，res = {tempFilePath:'文件的临时路径'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用 wx. saveFile，在小程序下次启动时才能访问得到。



2）示例代码 B-3

```
wx.downloadFile({
  url:'http://example.com/audio/123',//仅为示例,并非真实的资源
  success:function(res) {
    wx.playVoice({
      filePath:res.tempFilePath
    })
  }
})
```

B.1.3　Web Socket

1. wx.connectSocket(OBJECT)

创建一个 WebSocket 连接。一个微信小程序同时只能有一个 WebSocket 连接，如果当前已存在一个 WebSocket 连接，会自动关闭该连接，并重新创建一个 WebSocket 连接。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-14 所示。

表 B-14　wx.connectSocket（OBJECT）中 OBJECT 参数说明

参　数	类　型	必　填	说　明
url	String	是	开发者服务器接口地址，必须是 wss 协议，且域名必须是后台配置的合法域名
data	Object	否	请求的数据
header	Object	否	HTTP Header，header 中不能设置 Referer
method	String	否	默认是 GET，有效值为：OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-4

```
wx.connectSocket({
  url:'test.php',
  data:{
    x:'',
    y:''
  },
  header:{
'content-type':'application/json'
  },
```

```
    method:"GET"
})
```

2. wx.onSocketOpen(CALLBACK)

监听 WebSocket 连接打开事件，示例代码 B-5 如下所示。

```
wx.connectSocket({
  url:'test.php'
})
wx.onSocketOpen(function(res){
console.log('WebSocket 连接已打开!')
})
```

3. wx.onSocketError(CALLBACK)

监听 WebSocket 错误，示例代码 B-6 如下所示。

```
wx.connectSocket({
  url:'test.php'
})
wx.onSocketOpen(function(res){
console.log('WebSocket 连接已打开!')
})
wx.onSocketError(function(res){
console.log('WebSocket 连接打开失败,请检查!')
})
```

4. wx.sendSocketMessage(OBJECT)

通过 WebSocket 连接发送数据，需要先 wx.connectSocket，并在 wx.onSocketOpen 回调之后才能发送。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-15 所示。

表 B-15　wx.sendSocketMessage(OBJECT)中 OBJECT 参数说明

参　数	类　型	必填	说　明
data	String/ArrayBuffer	是	需要发送的内容
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-7

```
var socketOpen = false
var socketMsgQueue = [ ]
wx. connectSocket( {
  url:'test. php'
} )

wx. onSocketOpen( function( res) {
  socketOpen  = true
for ( var i = 0;i < socketMsgQueue. length;i ++ ) {
    sendSocketMessage( socketMsgQueue[ i] )
  }
  socketMsgQueue = [ ]
} )

function sendSocketMessage( msg) {
if ( socketOpen) {
    wx. sendSocketMessage( {
      data:msg
    } )
  } else {
    socketMsgQueue. push( msg)
  }
}
```

5. wx. onSocketMessage(CALLBACK)

监听 WebSocket 接受到服务器的消息事件。

1）CALLBACK 返回参数说明

CALLBACK 返回参数说明如表 B-16 所示。

表 B-16　wx. onSocketMessage(CALLBACK) 中 CALLBACK 返回参数说明

参　数	类　型	说　明
data	String/ArrayBuffer	服务器返回的消息

2）示例代码 B-8

```
wx. connectSocket( {
  url:'test. php'
} )

wx. onSocketMessage( function( res) {
console. log('收到服务器内容:' + res. data)
} )
```

6. wx. closeSocket( )

关闭 WebSocket 连接。

7. wx. onSocketClose(CALLBACK)

监听 WebSocket 关闭，示例代码 B-9 如下所示。

```
wx. connectSocket( {
  url:'test. php '
} )

//注意这里有时序问题,
//如果 wx. connectSocket 还没回调 wx. onSocketOpen,而先调用 wx. closeSocket,那么就做不到关闭
WebSocket 的目的。
//必须在 WebSocket 打开期间调用 wx. closeSocket 才能关闭。
wx. onSocketOpen( function( ) {
  wx. closeSocket( )
} )

wx. onSocketClose( function( res) {
console. log( 'WebSocket 已关闭! ' )
} )
```

B. 2　媒体

B. 2. 1　图片

1. wx. chooseImage(OBJECT)

从本地相册选择图片或使用相机拍照。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-17 所示。

表 B-17　wx. chooseImage(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
count	Number	否	最多可以选择的图片张数，默认 9
sizeType	StringArray	否	original 原图，compressed 压缩图，默认二者都有
sourceType	StringArray	否	album 从相册选图，camera 使用相机，默认二者都有

续表

参　数	类　型	必　填	说　明
success	Function	是	成功则返回图片的本地文件路径列表 tempFilePaths
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用 wx. saveFile，在小程序下次启动时才能访问得到。

2）示例代码 B-10

```
wx. chooseImage({
  count:1,                                    // 默认 9
  sizeType: ['original', 'compressed'],       // 可以指定是原图还是压缩图,默认二者都有
  sourceType: ['album', 'camera'],            // 可以指定来源是相册还是相机,默认二者都有
  success:function (res){
// 返回选定照片的本地文件路径列表,tempFilePath 可以作为 img 标签的 src 属性显示图片
var tempFilePaths = res. tempFilePaths
  }
})
```

2. wx. previewImage(OBJECT)

预览图片。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-18 所示。

表 B-18　wx. previewImage(OBJECT)中 OBJEC 参数说明

参　数	类　型	必　填	说　明
current	String	否	当前显示图片的链接，不填则默认为 urls 的第一张
urls	StringArray	是	需要预览的图片链接列表
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-11

```
wx. previewImage({
  current:'',                 // 当前显示图片的 http 链接
  urls: []                    // 需要预览的图片 http 链接列表
})
```

3. wx. getImageInfo(OBJECT)

获取图片信息。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-19 所示。

表 B-19　wx. getImageInfo(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
src	String	是	图片的路径、可以是相对路径、临时文件路径、存储文件路径、网络图片路径
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

success 返回参数说明如表 B-20 所示。

表 B-20　wx. getImageInfo(OBJECT)的 success 返回参数说明

参　数	类　型	说　明
width	Number	图片宽度，单位 px
height	Number	图片高度，单位 px

2）示例代码 B-12

```
wx.getImageInfo({
  src:'images/a.jpg',
  success:function(res){
console.log(res.width)
console.log(res.height)
  }
})

wx.chooseImage({
  success:function(res){
    wx.getImageInfo({
      src: res.tempFilePaths[0],
      success:function(res){
console.log(res.width)
console.log(res.height)
      }
    })
  }
})
```

B.2.2　录音

1. wx.startRecord(OBJECT)

开始录音。当主动调用 wx.stopRecord，或者录音超过 1 分钟时自动结束录音，返回录

音文件的临时文件路径。当用户离开小程序时，此接口无法调用。

OBJECT 参数说明如表 B-21 所示。

表 B-21　wx. startRecord(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	否	录音成功后调用，返回录音文件的临时文件路径，res = {tempFilePath: '录音文件的临时路径'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用 wx. saveFile，在小程序下次启动时才能访问得到。

2. wx. stopRecord( )

主动调用停止录音。示例代码 B-13 如下所示。

```
wx. startRecord({
  success:function(res){
    var tempFilePath = res. tempFilePath
  },
  fail:function(res){
    //录音失败
  }
})
setTimeout(function(){
  //结束录音
  wx. stopRecord()
},10000)
```

3. Bug & Tip

(1) Tip:wx. startRecord 接口需要用户授权，请兼容用户拒绝授权的场景。

B. 2. 3　音频播放控制

1. wx. playVoice(OBJECT)

开始播放语音，同时只允许一个语音文件正在播放，如果前一个语音文件还没播放完，将中断前一个语音播放。

1) OBJECT 参数说明

OBJECT 参数说明如表 B-22 所示。

表 B-22　wx. playVoice(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
filePath	String	是	需要播放的语音文件的文件路径
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-14

```
wx. startRecord({
  success:function(res){
var tempFilePath = res. tempFilePath
    wx. playVoice({
      filePath: tempFilePath,
      complete:function(){
      }
    })
  }
})
```

2. wx. pauseVoice()

暂停正在播放的语音。再次调用 wx. playVoice 播放同一个文件时，会从暂停处开始播放。如果想从头开始播放，需要先调用 wx. stopVoice。示例代码 B-15 如下所示。

```
wx. startRecord({
  success:function(res){
var tempFilePath = res. tempFilePath
      wx. playVoice({
      filePath: tempFilePath
    })

    setTimeout(function(){
//暂停播放
      wx. pauseVoice()
    },5000)
  }
})
```

3. wx. stopVoice()

结束播放语音，示例代码 B-16 如下所示。

```
wx.startRecord({
  success:function(res){
var tempFilePath = res.tempFilePath
    wx.playVoice({
      filePath:tempFilePath
    })

    setTimeout(function(){
      wx.stopVoice()
    },5000)
  }
})
```

B.2.4　音乐播放控制

1. wx.getBackgroundAudioPlayerState(OBJECT)

获取音乐播放状态。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-23 所示。

表 B-23　wx.getBackgroundAudioPlayerState(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-24 所示。

表 B-24　wx.getBackgroundAudioPlayerState(OBJECT) success 返回参数说明

参　数	说　明
duration	选定音频的长度（单位：s），只有在当前有音乐播放时返回
currentPosition	选定音频的播放位置（单位：s），只有在当前有音乐播放时返回
status	播放状态（2：没有音乐在播放，1：播放中，0：暂停中）
downloadPercent	音频的下载进度（整数，80 代表 80%），只有在当前有音乐播放时返回
dataUrl	歌曲数据链接，只有在当前有音乐播放时返回

3）示例代码 B-17

```
wx. getBackgroundAudioPlayerState( {
        success:function( res) {
var status = res. status
var dataUrl = res. dataUrl
var currentPosition = res. currentPosition
var duration = res. duration
var downloadPercent = res. downloadPercent
        }
})
```

2. wx. playBackgroundAudio(OBJECT)

播放音乐，同时只能有一首音乐正在播放。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-25 所示。

表 B-25　wx. playBackgroundAudio(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
dataUrl	String	是	音乐链接
title	String	否	音乐标题
coverImgUrl	String	否	封面 URL
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-18

```
wx. playBackgroundAudio( {
    dataUrl:'',
    title:'',
    coverImgUrl:''
})
```

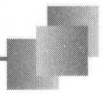
3. wx. pauseBackgroundAudio( )

暂停播放音乐，示例代码 B-19 如下所示。

```
wx. pauseBackgroundAudio( )
```

4. wx. seekBackgroundAudio(OBJECT)

控制音乐播放进度。



1）OBJECT 参数说明

OBJECT 参数说明如表 B-26 所示。

表 B-26　wx. seekBackgroundAudio(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
position	Number	是	音乐位置，单位：秒
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-20

```
wx. seekBackgroundAudio({
    position:30
})
```

5. wx. stopBackgroundAudio()

停止播放音乐，示例代码 B-21 如下所示。

```
wx. stopBackgroundAudio()
```

6. wx. onBackgroundAudioPlay(CALLBACK)

监听音乐播放。

7. wx. onBackgroundAudioPause(CALLBACK)

监听音乐暂停。

8. wx. onBackgroundAudioStop(CALLBACK)

监听音乐停止。

9. Bug & Tip

bug：iOS 6. 3. 30 wx. seekBackgroundAudio 会有短暂延迟；

B. 2. 5　音频组件控制

1. wx. createAudioContext(audioId)

创建并返回 audio 上下文 audioContext 对象。

2. audioContext

audioContext 通过 audioId 跟一个 <audio/> 组件绑定，通过它可以操作对应的 <audio/> 组件。audioContext 对象的方法列表如表 B-27 所示。

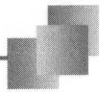
表 B-27　audioContext 对象的方法列表

方　法	参　数	说　明
setSrc	src	音频的地址
play	无	播放
pause	无	暂停
seek	position	跳转到指定位置，单位 s

3. 示例代码 B-22

```
<!-- audio.wxml -->
<audio  src="{{src}}" id="myAudio" ></audio>

<button type="primary" bindtap="audioPlay">播放</button>
<button type="primary" bindtap="audioPause">暂停</button>
<button type="primary" bindtap="audio14">设置当前播放时间为 14 秒</button>
<button type="primary" bindtap="audioStart">回到开头</button>
// audio.js
Page({
  onReady:function(e){
    // 使用 wx.createAudioContext 获取 audio 上下文 context
    this.audioCtx=wx.createAudioContext('myAudio')
this.audioCtx.setSrc('http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?guid=ffffffff
82def4af4b12b3cd9337d5e7&uin=346897220&vkey=6292F51E1E384E06DCBDC9AB7C49FD713D6
32D313AC4858BACB8DDD29067D3C601481D36E62053BF8DFEAF74C0A5CCFADD6471160CAF3E6
A&fromtag=46')
    this.audioCtx.play()
  },
  data:{
    src:''
  },
  audioPlay:function(){
    this.audioCtx.play()
  },
  audioPause:function(){
    this.audioCtx.pause()
  },
```



```
    audio14:function (){
      this. audioCtx. seek(14)
    },
    audioStart:function (){
      this. audioCtx. seek(0)
    }
  })
```

B. 2. 6　视频

1. wx. chooseVideo(OBJECT)

拍摄视频或从手机相册中选视频，返回视频的临时文件路径。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-28 所示。

表 B-28　wx. chooseVideo(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
sourceType	StringArray	否	album 从相册选视频，camera 使用相机拍摄，默认为：['album', 'camera']
maxDuration	Number	否	拍摄视频最长拍摄时间，单位为 s。最长支持 60s
camera	String	否	默认调起的为前置还是后置摄像头。front：前置，back：后置，默认 back
success	Function	否	接口调用成功，返回视频文件的临时文件路径，详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）返回参数说明

返回参数说明如表 B-29 所示。

表 B-29　wx. chooseVideo(OBJECT)返回参数说明

参　数	说　明
tempFilePath	选定视频的临时文件路径
duration	选定视频的时间长度
size	选定视频的数据量大小
height	返回选定视频的长
width	返回选定视频的宽

注：文件的临时路径，在小程序本次启动期间可以正常使用，如需持久保存，需在主动调用 wx. saveFile，在小程序下次启动时才能访问得到。

2. 示例代码 B-23

```
<view class = "container" >
    <video src = "{{src}}" > </video >
    <button bindtap = "bindButtonTap" > 获取视频 </button >
</view >
Page({
    bindButtonTap:function(){
        var that = this
        wx. chooseVideo({
            sourceType: ['album','camera'],
            maxDuration:60,
        camera:'back',
            success:function(res){
                that. setData({
                    src: res. tempFilePath
                })
            }
        })
    }
})
```

B. 2. 7　视频组件控制

1. wx. createVideoContext(videoId)

创建并返回 video 上下文 videoContext 对象。

2. videoContext

videoContext 通过 videoId 跟一个 video 组件绑定，通过它可以操作一个 video 组件。videoContext对象的方法列表如表 B-30 所示。

表 B-30　videoContext 对象的方法列表

方　法	参　数	说　明
play	无	播放
pause	无	暂停
seek	position	跳转到指定位置，单位为 s
sendDanmu	danmu	发送弹幕，danmu 包含两个属性：text、color

3. 示例代码 B-24

```
<view class = "section tc" >
    <video id = "myVideo"
src = " http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload? filekey = 30280201010421301f0
201690402534804102ca905ce620b1241b726bc41dcff44e00204012882540400&bizid = 1023&hy = SH&
fileparam = 302c020101042530230204136ffd93020457e3c4ff02024ef202031e8d7f02030f42400204045a
320a0201000400"    enable - danmu danmu - btn controls > </video >
    <view class = "btn - area" >
      <input bindblur = "bindInputBlur"/ >
      <button bindtap = "bindSendDanmu" > 发送弹幕 </button >
    </view >
</view >
function getRandomColor ( ) {
  let rgb = [ ]
  for (let i = 0 ; i < 3; ++ i) {
    let color = Math. floor( Math. random( ) * 256). toString(16)
    color = color. length = = 1 ? '0 ' + color : color
    rgb. push( color)
  }
  return '#' + rgb. join( '')
}

Page( {
  onReady: function (res) {
    this. videoContext = wx. createVideoContext( 'myVideo ')
  },
  inputValue: '',
  bindInputBlur: function( e) {
    this. inputValue = e. detail. value
  },
  bindSendDanmu: function ( ) {
    this. videoContext. sendDanmu( {
      text: this. inputValue,
      color: getRandomColor( )
    } )
  }
} )
```

B.3 文件

1. wx.saveFile(OBJECT)

保存文件到本地。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-31 所示。

表 B-31　wx.saveFile(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
tempFilePath	String	是	需要保存的文件的临时路径
success	Function	否	返回文件的保存路径，res = {savedFilePath: '文件的保存路径'}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-25

```
wx.startRecord({
  success:function(res){
var tempFilePath = res.tempFilePath
    wx.saveFile({
      tempFilePath: tempFilePath,
      success:function(res){
var savedFilePath = res.savedFilePath
      }
    })
  }
})
```

3）Bug & Tip

Tip：本地文件存储的大小限制为 100M。

2. wx.getSavedFileList(OBJECT)

获取本地已保存的文件列表

1）OBJECT 参数说明

OBJECT 参数说明如表 B-32 所示。

表 B-32　wx.getSavedFileList(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数，返回结果见 success 返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-33 所示。

表 B-33　wx. getSavedFileList(OBJECT) success 返回参数说明

参　数	类　型	说　明
errMsg	String	接口调用结果
fileList	Object Array	文件列表

3）fileList 中的项目说明

fileList 中的项目说明如表 B-34 所示。

表 B-34　fileList 中的项目说明

键	类　型	说　明
filePath	String	文件的本地路径
createTime	Number	文件保存时的时间戳，从 1970/01/01 08:00:00 到当前时间的秒数
size	Number	文件大小，单位为 B

4）示例代码 B-26

```
wx. getSavedFileList({
  success:function(res){
console. log(res. fileList)
  }
})
```

3. wx. getSavedFileInfo(OBJECT)

获取本地文件的文件信息

1）OBJECT 参数说明

OBJECT 参数说明如表 B-35 所示。

表 B-35　wx. getSavedFileInfo(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
filePath	String	是	文件路径
success	Function	否	接口调用成功的回调函数，返回结果见 success 返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-36 所示。

表 B-36　wx. getSavedFileInfo(OBJECT) success 返回参数说明

参　数	类　型	说　明
errMsg	String	接口调用结果
size	Number	文件大小，单位为 B
createTime	Number	文件保存时的时间戳，从 1970/01/01 08:00:00 到当前时间的秒数

3）示例代码 B-27

```
wx. getSavedFileInfo({
  filePath:'wxfile://somefile', //仅做示例用,非真正的文件路径
  success:function(res){
console. log(res. size)
console. log(res. createTime)
  }
})
```

4. wx. removeSavedFile(OBJECT)

删除本地存储的文件

1）OBJECT 参数说明

OBJECT 参数说明如表 B-37 所示。

表 B-37　wx. getSavedFileList(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
filePath	String	是	需要删除的文件路径
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-28

```
wx. getSavedFileList({
  success:function(res){
if (res. fileList. length >0){
      wx. removeSavedFile({
        filePath: res. fileList[0]. filePath,
        complete:function(res){
console. log(res)
        }
      })
    }
  }
})
```



5. wx. openDocument(OBJECT)

新开页面打开文档，支持格式：doc、xls、ppt、pdf、docx、xlsx、pptx。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-38 所示。

表 B-38　wx. openDocument(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
filePath	String	是	文件路径，可通过 downFile 获得
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-29

```
wx. downloadFile({
  url:'http://example. com/somefile. pdf',
  success:function (res){
var filePath = res. tempFilePath
    wx. openDocument({
      filePath: filePath,
      success:function (res){
console. log('打开文档成功')
      }
    })
  }
})
```

B. 4　数据缓存

每个微信小程序都可以有自己的本地缓存，可以通过 wx. setStorage(wx. setStorageSync)、wx. getStorage(wx. getStorageSync)、wx. clearStorage(wx. clearStorageSync)对本地缓存进行设置、获取和清理。本地缓存最大为 10MB。

注： localStorage 是永久存储的，但是我们不建议将关键信息全部存在 localStorage，以防用户换设备的情况。

1. wx. setStorage(OBJECT)

将数据存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个异步接口。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-39 所示。

表 B-39　wx. setStorage(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中指定的 key
data	Object/String	是	需要存储的内容
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-30

```
wx. setStorage({
  key:"key"
  data:"value"
})
```

2. wx. setStorageSync(KEY,DATA)

将 data 存储在本地缓存中指定的 key 中，会覆盖掉原来该 key 对应的内容，这是一个同步接口。

1）参数说明

参数说明如表 B-40 所示。

表 B-40　wx. setStorageSync(KEY,DATA)中参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中指定的 key
data	Object/String	是	需要存储的内容

2）示例代码 B-31

```
try{
    wx. setStorageSync('key', 'value')
}catch (e){
}
```

3. wx. getStorage(OBJECT)

从本地缓存中异步获取指定 key 对应的内容。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-41 所示。

表 B-41　wx. getStorage(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中指定的 key
success	Function	是	接口调用的回调函数，res = {data: key 对应的内容}
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-32

```
wx. getStorage({
  key:'key',
  success:function(res){
console. log(res. data)
  }
})
```

4. wx. getStorageSync(KEY)

从本地缓存中同步获取指定 key 对应的内容。

1）参数说明

参数说明如表 B-42 所示。

表 B-42　wx. getStorageSync(KEY)中参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中的指定的 key

2）示例代码 B-33

```
try{
var value = wx. getStorageSync('key')
if (value){
// Do something with return value
  }
}catch (e){
// Do something when catch error
}
```

5. wx. getStorageInfo(OBJECT)

异步获取当前 storage 的相关信息。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-43 所示。

表 B-43　wx. getStorageInfo(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	是	接口调用的回调函数，详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-44 所示。

表 B-44　wx. getStorageInfo success 返回参数说明

参　数	类　型	说　明
keys	String Array	当前 storage 中所有的 key
currentSize	Number	当前占用的空间大小，单位为 kb
limitSize	Number	限制的空间大小，单位为 kb

3）示例代码 B-34

```
wx. getStorageInfo({
   success:function(res){
console. log(res. keys)
console. log(res. currentSize)
console. log(res. limitSize)
   }
})
```

6. wx. getStorageInfoSync

同步获取当前 storage 的相关信息，示例代码 B-35 如下所示。

```
try{
var res = wx. getStorageInfoSync()
console. log(res. keys)
console. log(res. currentSize)
console. log(res. limitSize)
}catch (e){
// Do something when catch error
}
```

7. wx. removeStorage(OBJECT)

从本地缓存中异步移除指定 key。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-45 所示。

表 B-45　wx. removeStorage(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中指定的 key
success	Function	是	接口调用的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-36

```
wx. removeStorage({
  key:'key',
  success:function(res){
console. log(res. data)
  }
})
```

8. wx. removeStorageSync(KEY)

从本地缓存中同步移除指定 key。

1）参数说明

参数说明如表 B-46 所示。

表 B-46　wx. removeStorageSync(KEY)参数说明

参　数	类　型	必　填	说　明
key	String	是	本地缓存中的指定的 key

2）示例代码 B-37

```
try{
  wx. removeStorageSync('key')
}catch (e){
// Do something when catch error
}
```

9. wx. clearStorage()

清理本地数据缓存，示例代码 B-38 如下所示。

```
wx. clearStorage()
```

10. wx. clearStorageSync()

同步清理本地数据缓存。示例代码 B-39 如下所示。

```
try{
    wx. clearStorageSync( )
}catch(e){
// Do something when catch error
}
```

11. Bug & Tip

Tip：本地数据存储的大小限制为5M。

B.5 位置

B.5.1 获取位置

1. wx. getLocation(OBJECT)

获取当前的地理位置、速度。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-47 所示。

表 B-47 wx. getLocation(OBJECT)中 OBJECT 参数说明

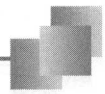
参 数	类 型	必 填	说 明
type	String	否	默认为 wgs84 返回 gps 坐标，gcj02 返回可用于 wx. openLocation 的坐标
success	Function	是	接口调用成功的回调函数，返回内容详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-48 所示。

表 B-48 wx. getLocation(OBJECT) success 返回参数说明

参 数	说 明
latitude	纬度，浮点数，范围为 -90 ~ 90，负数表示南纬
longitude	经度，浮点数，范围为 -180 ~ 180，负数表示西经
speed	速度，浮点数，单位 m/s
accuracy	位置的精确度



3）示例代码 B-40

```
wx. getLocation({
  type:'wgs84',
  success:function(res){
var latitude = res. latitude
var longitude = res. longitude
var speed = res. speed
var accuracy = res. accuracy
  }
})
```

2. wx. chooseLocation(OBJECT)

打开地图选择位置。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-49 所示。

表 B-49　wx. chooseLocation(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	是	接口调用成功的回调函数，返回内容详见返回参数说明
cancel	Function	否	用户取消时调用
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-50 所示。

表 B-50　wx. chooseLocation(OBJECT) success 返回参数说明

参　数	说　明
name	位置名称
address	详细地址
latitude	纬度，浮点数，范围为 -90 ~ 90，负数表示南纬
longitude	经度，浮点数，范围为 -180 ~ 180，负数表示西经

B. 5. 2　查看位置

1. wx. openLocation(OBJECT)

使用微信内置地图查看位置。

1）OBEJCT 参数说明

OBEJCT 参数说明如表 B-51 所示。

表 B-51　wx. openLocation(OBJECT)中 OBEJCT 参数说明

参　数	类　型	必　填	说　明
latitude	Float	是	纬度，范围为 -90 ~ 90，负数表示南纬
longitude	Float	是	经度，范围为 -180 ~ 180，负数表示西经
scale	INT	否	缩放比例，范围 1 ~ 28，默认为 28
name	String	否	位置名
address	String	否	地址的详细说明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-41

```
wx. getLocation({
  type:'gcj02 ', //返回可以用于 wx. openLocation 的经纬度
  success:function(res){
var latitude = res. latitude
var longitude = res. longitude
    wx. openLocation({
      latitude: latitude,
      longitude: longitude,
      scale:28
    })
  }
})
```

2. Bug & Tip

Bug：iOS 6. 3. 30 type 参数不生效，只会返回 wgs84 类型的坐标信息。

B. 5. 3　地图组件控制

1. wx. createMapContext(mapId)

创建并返回 map 上下文 mapContext 对象。

2. mapContext

mapContext 通过 mapId 跟一个 <map/> 组件绑定，通过它可以操作对应的 <map/> 组件。

1）mapContext 对象的方法列表

mapContext 对象的方法列表如表 B-52 所示。

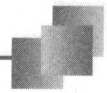


表 B-52　mapContext 对象的方法列表

参　数	类　型	说　明
getCenterLocation	OBJECT	获取当前地图中心的经纬度，返回的是 gcj02 坐标系，可以用于 wx. openLocation
moveToLocation	无	将地图中心移动到当前定位点，需要配合 map 组件的 show - location 使用

2）getCenterLocation 的 OBJECT 参数列表

getCenterLocation 的 OBJECT 参数列表如表 B-53 所示。

表 B-53　getCenterLocation 的 OBJECT 参数列表

参　数	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数，res = { longitude: "经度", latitude: "纬度" }
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

3）示例代码 B-42

```
<!-- map. wxml -->
<map id = "myMap" show - location / >

<button type = "primary" bindtap = "getCenterLocation" > 获取位置 </button >
<button type = "primary" bindtap = "moveToLocation" > 移动位置 </button >
// map. js
Page({
  onReady:function (e){
    // 使用 wx. createMapContext 获取 map 上下文
    this. mapCtx = wx. createMapContext('myMap')
  },
  getCenterLocation:function (){
    this. mapCtx. getCenterLocation({
      success:function(res){
        console. log(res. longitude)
        console. log(res. latitude)
      }
    })
  },
  moveToLocation:function (){
    this. mapCtx. moveToLocation()
  }
})
```

B.6　设备

B.6.1　系统信息

1. wx.getSystemInfo(OBJECT)

获取系统信息。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-54 所示。

表 B-54　wx.getSystemInfo(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	是	接口调用成功的回调
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 回调参数说明

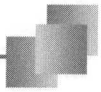
success 回调参数说明如表 B-55 所示。

表 B-55　wx.getSystemInfo(OBJECT) success 回调参数说明

属　性	说　明
model	手机型号
pixelRatio	设备像素比
windowWidth	窗口宽度
windowHeight	窗口高度
language	微信设置的语言
version	微信版本号

3）示例代码 B-43

```
wx.getSystemInfo({
  success:function(res){
console.log(res.model)
console.log(res.pixelRatio)
console.log(res.windowWidth)
console.log(res.windowHeight)
console.log(res.language)
console.log(res.version)
  }
})
```



2. wx. getSystemInfoSync( )

获取系统信息同步接口，示例代码 B-44 如下所示。

```
try{
var res = wx. getSystemInfoSync( )
console. log( res. model)
console. log( res. pixelRatio)
console. log( res. windowWidth)
console. log( res. windowHeight)
console. log( res. language)
console. log( res. version)
}catch (e){
// Do something when catch error
}
```

B. 6. 2　网络状态

wx. getNetworkType(OBJECT)，获取网络类型。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-56 所示。

表 B-56　wx. getNetworkType(OBJECT) 中 OBJECT 参数说明

参　数	类　型	必　填	说　明
success	Function	是	接口调用成功，返回网络类型 networkType
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-45

```
wx. getNetworkType({
  success:function(res){
var networkType = res. networkType          //返回网络类型 2g,3g,4g,wifi
  }
})
```

B. 6. 3　重力感应

wx. onAccelerometerChange(CALLBACK)，监听重力感应数据，频率为 5 次/秒。

1）CALLBACK 返回参数

CALLBACK 返回参数如表 B-57 所示。

表 B-57　wx. onAccelerometerChange(CALLBACK)中 CALLBACK 返回参数

参　数	类　型	说　明
x	Number	X 轴
y	Number	Y 轴
z	Number	Z 轴

2）示例代码 B-46

```
wx. onAccelerometerChange(function(res){
console. log(res. x)
console. log(res. y)
console. log(res. z)
})
```

B. 6. 4　罗盘

wx. onCompassChange(CALLBACK)，监听罗盘数据，频率为 5 次/秒。

1）CALLBACK 返回参数

CALLBACK 返回参数如表 B-58 所示。

表 B-58　wx. onCompassChange(CALLBACK)中 CALLBACK 返回参数

参　数	类　型	说　明
direction	Number	面对的方向度数

2）示例代码 B-47

```
wx. onCompassChange(function (res){
console. log(res. direction)
})
```

B. 6. 5　拨打电话

wx. makePhoneCall(OBJECT)，拨打电话。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-59 所示。

表 B-59　wx. makePhoneCall(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
phoneNumber	String	是	需要拨打的电话号码
success	Function	否	接口调用成功的回调
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-48

```
wx. makePhoneCall({
  phoneNumber:'1340000 '//仅为示例,并非真实的电话号码
})
```

B. 6. 6　扫码

wx. scanCode(OBJECT)，调起客户端扫码界面，扫码成功后返回对应的结果。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-60 所示。

表 B-60　wx. scanCode(OBJECT)中 OBJECT 参数说明

参数	类型	必填	说　明
success	Function	否	接口调用成功的回调函数，返回内容详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-61 所示。

表 B-61　wx. scanCode(OBJECT) success 返回参数说明

参　数	说　明
result	码的内容

3）示例代码 B-49

```
wx. scanCode({
  success:(res) =>{
    console. log(res)
  }
})
```

B. 7　界面

B. 7. 1　交互反馈

1. wx. showToast(OBJECT)

显示消息提示框。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-62 所示。

表 B-62　wx. showToast(OBJECT)中 OBJECT 参数说明

参　数	类型	必填	说　明
title	String	是	提示的内容
icon	String	否	图标，只支持"success"、"loading"
duration	Number	否	提示的延迟时间，单位毫秒，默认：1500，最大为 10000
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-50

```
wx. showToast({
  title:'成功',
  icon:'success',
  duration:2000
})
```

2. wx. hideToast()

隐藏消息提示框，示例代码 B-51 如下所示。

```
wx. showToast({
  title:'加载中',
  icon:'loading',
  duration:10000
})

setTimeout(function(){
  wx. hideToast()
},2000)
```

3. wx. showModal(OBJECT)

显示模态弹窗。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-63 所示。

表 B-63　wx. showModal(OBJECT)中 OBJECT 参数说明

参　数	类　型	必　填	说　明
title	String	是	提示的标题
content	String	是	提示的内容
showCancel	Boolean	否	是否显示取消按钮，默认为 true

续表

参　数	类　型	必　填	说　明
cancelText	String	否	取消按钮的文字，默认为"取消"，最多 4 个字符
cancelColor	HexColor	否	取消按钮的文字颜色，默认为"#000000"
confirmText	String	否	确定按钮的文字，默认为"确定"，最多 4 个字符
confirmColor	HexColor	否	确定按钮的文字颜色，默认为"#3CC51F"
success	Function	否	接口调用成功的回调函数，返回 res. confirm 为 true 时，表示用户点击确定按钮
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-52

```
wx. showModal({
  title:'提示',
  content:'这是一个模态弹窗',
  success:function(res){
if (res. confirm){
console. log('用户点击确定')
    }
  }
})
```

4. wx. showActionSheet(OBJECT)

显示操作菜单。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-64 所示。

表 B-64　wx. showActionSheet(OBJECT)中 OBJECT 参数说明

参　数	类型	必填	说　明
itemList	String Array	是	按钮的文字数组，数组长度最大为 6 个
itemColor	HexColor	否	按钮的文字颜色，默认为"#000000"
success	Function	否	接口调用成功的回调函数，详见返回参数说明
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-65 所示。

表 B-65　wx. showActionSheet(OBJECT) success 返回参数说明

参　数	类型	说　明
cancel	Boolean	用户是否取消选择
tapIndex	Number	用户单击的按钮，从上到下的顺序，从 0 开始

3）示例代码 B-53

```
wx. showActionSheet({
  itemList:['A','B','C'],
  success:function(res){
if (! res. cancel){
console. log(res. tapIndex)
    }
  }
})
```

5. Bug & Tip

bug：Android 6. 3. 30，wx. showModal 的返回的 confirm 一直为 true。

B. 7. 2　设置导航条

1. wx. setNavigationBarTitle(OBJECT)

动态设置当前页面的标题。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-66 所示。

表 B-66　wx. setNavigationBarTitle(OBJECT) 中 OBJECT 参数说明

参　数	类型	必填	说　明
title	String	是	页面标题
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-54

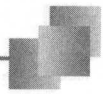
```
wx. setNavigationBarTitle({
  title:'当前页面'
})
```

2. wx. showNavigationBarLoading()

在当前页面显示导航条加载动画。

3. wx. hideNavigationBarLoading()

隐藏导航条加载动画。



B.7.3　导航

1. wx.navigateTo(OBJECT)

保留当前页面，跳转到应用内的某个页面，使用 wx.navigateBack 可以返回到原页面。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-67 所示。

表 B-67　wx.navigateTo(OBJECT)中 OBJECT 参数说明

参　数	类型	必填	说　明
url	String	是	需要跳转的应用内页面的路径，路径后可以带参数。参数与路径之间使用？分隔，参数键与参数值用 = 相连，不同参数用 & 分隔，如'path？key = value&key2 = value2'
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-55

```
wx.navigateTo({
  url:'test？id=1'
})
//test.js
Page({
  onLoad:function(option){
console.log(option.query)
  }
})
```

注：为了不让用户在使用小程序时造成困扰，我们规定页面路径只能是五层，请尽量避免多层级的交互方式。

2. wx.redirectTo(OBJECT)

关闭当前页面，跳转到应用内的某个页面。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-68 所示。

表 B-68　wx.redirectTo(OBJECT)中 OBJECT 参数说明

参　数	类型	必填	说　明
url	String	是	需要跳转的应用内页面的路径
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-56

```
wx.redirectTo({
  url:'test? id=1'
})
```

3. wx.navigateBack(OBJECT)

关闭当前页面，返回上一页面或多级页面。可通过 getCurrentPages()获取当前的页面栈，决定需要返回几层。

OBJECT 参数说明如表 B-69 所示。

表 B-69　wx.navigateBack(OBJECT)中 OBJECT 参数说明

参数	类型	默认值	说　明
delta	Number	1	返回的页面数，如果 delta 大于现有页面数，则返回到首页

B.7.4　动画

1. wx.createAnimation(OBJECT)

创建一个动画实例 animation。调用实例的方法来描述动画。最后通过动画实例的 export 方法导出动画数据传递给组件的 animation 属性。

注： *export 方法每次调用后会清掉之前的动画操作。*

1）OBJECT 参数说明

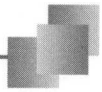
OBJECT 参数说明如表 B-70 所示。

表 B-70　wx.createAnimation(OBJECT)中 OBJECT 参数说明

参　数	类型	必填	说　明
duration	Integer	否	动画持续时间，单位 ms，默认值 400
timingFunction	String	否	定义动画的效果，默认值"linear"，有效值："linear"、"ease"、"ease-in"、"ease-in-out"、"ease-out"、"step-start"、"step-end"
delay	Integer	否	动画延迟时间，单位 ms，默认值 0
transformOrigin	String	否	设置 transform-origin，默认为"50% 50% 0"

2）示例代码 B-57

```
var animation = wx.createAnimation({
  transformOrigin:"50% 50%",
  duration:1000,
  timingFunction:"ease",
  delay:0
})
```



2. animation

动画实例可以调用以下方法来描述动画，调用结束后会返回自身，支持链式调用的写法。

1）样式

样式如表 B-71 所示。

表 B-71　样式

方　法	参　数	说　明
opacity	value	透明度，参数范围 0 ~ 1
backgroundColor	color	颜色值
width	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
height	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
top	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
left	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
bottom	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值
right	length	长度值，如果传入 Number 则默认使用 px，可传入其他自定义单位的长度值

2）旋转

旋转如表 B-72 所示。

表 B-72　旋转

方　法	参　数	说　明
rotate	deg	deg 的范围 -180 ~ 180，从原点顺时针旋转一个 deg 角度
rotateX	deg	deg 的范围 -180 ~ 180，在 *X* 轴旋转一个 deg 角度
rotateY	deg	deg 的范围 -180 ~ 180，在 *Y* 轴旋转一个 deg 角度
rotateZ	deg	deg 的范围 -180 ~ 180，在 *Z* 轴旋转一个 deg 角度
rotate3d	(x,y,z,deg)	同 transform - function rotate3d

3）缩放

缩放如表 B-73 所示。

表 B-73　缩放

方　法	参　数	说　明
scale	sx，[sy]	一个参数时，表示在 *X* 轴、*Y* 轴同时缩放 sx 倍数；两个参数时表示在 *X* 轴缩放 sx 倍数，在 *Y* 轴缩放 sy 倍数
scaleX	sx	在 *X* 轴缩放 sx 倍数
scaleY	sy	在 *Y* 轴缩放 sy 倍数
scaleZ	sz	在 *Z* 轴缩放 sy 倍数
scale3d	(sx,sy,sz)	在 *X* 轴缩放 sx 倍数，在 *Y* 轴缩放 sy 倍数，在 *Z* 轴缩放 sz 倍数

4）偏移

偏移如表 B-74 所示。

表 B-74　偏移

方　法	参　数	说　明
translate	tx，[ty]	一个参数时，表示在 X 轴偏移 tx，单位 px；两个参数时，表示在 X 轴偏移 tx，在 Y 轴偏移 ty，单位 px
translateX	tx	在 X 轴偏移 tx，单位 px
translateY	ty	在 Y 轴偏移 tx，单位 px
translateZ	tz	在 Z 轴偏移 tx，单位 px
translate3d	(tx,ty,tz)	在 X 轴偏移 tx，在 Y 轴偏移 ty，在 Z 轴偏移 tz，单位 px

5）倾斜

倾斜如表 B-75 所示。

表 B-75　倾斜

方　法	参　数	说　明
skew	ax，[ay]	参数范围 -180 ~ 180；一个参数时，Y 轴坐标不变，X 轴坐标延顺时针倾斜 ax 度；两个参数时，分别在 X 轴倾斜 ax 度，在 Y 轴倾斜 ay 度
skewX	ax	参数范围 -180 ~ 180；Y 轴坐标不变，X 轴坐标延顺时针倾斜 ax 度
skewY	ay	参数范围 -180 ~ 180；X 轴坐标不变，Y 轴坐标延顺时针倾斜 ay 度

6）矩阵变形

矩阵变形如表 B-76 所示。

表 B-76　矩阵变形

方　法	参　数	说　明
matrix	(a,b,c,d,tx,ty)	同 transform - function matrix
matrix3d		同 transform - function matrix3d

3. 动画队列

调用动画操作方法后要调用 step() 来表示一组动画完成，可以在一组动画中调用任意多个动画方法，一组动画中的所有动画会同时开始，一组动画完成后才会进行下一组动画。step 可以传入一个跟 wx. createAnimation() 一样的配置参数用于指定当前组动画的配置。示例代码 B-58 如下所示。

```
<view animation = "{{animationData}}" style = "background:red;height:100rpx;width:100rpx" >
</view >
Page({
  data:{
    animationData:{}
  },
  onShow:function(){
var animation = wx. createAnimation({
      duration:1000,
```

```
            timingFunction:'ease',
        })

    this.animation = animation

        animation.scale(2,2).rotate(45).step()

    this.setData({
            animationData:animation.export()
        })

        setTimeout(function(){
            animation.translate(30).step()
    this.setData({
                animationData:animation.export()
            })
        }.bind(this),1000)
    },
    rotateAndScale:function(){
    // 旋转同时放大
    this.animation.rotate(45).scale(2,2).step()
    this.setData({
            animationData:this.animation.export()
        })
    },
    rotateThenScale:function(){
    // 先旋转后放大
    this.animation.rotate(45).step()
    this.animation.scale(2,2).step()
    this.setData({
            animationData:this.animation.export()
        })
    },
    rotateAndScaleThenTranslate:function(){
    // 先旋转同时放大,然后平移
    this.animation.rotate(45).scale(2,2).step()
    this.animation.translate(100,100).step({duration:1000})
    this.setData({
            animationData:this.animation.export()
        })
    }
})
```

4. Bug & Tip

Bug：iOS/Android 6.3.30 通过 step（）分隔动画，只有第一步动画能生效。

B.7.5　绘图

绘图 API 接口如表 B-77 所示。

表 B-77　绘图 API 接口

方　法	说　明
createCanvasContext	创建 canvas 绘图上下文（指定 canvasId）
createContext（不推荐使用）	创建 canvas 绘图上下文
drawCanvas（不推荐使用）	进行绘图
canvasToTempFilePath	导出图片

B.8　开放接口

B.8.1　登录

1. wx.login(OBJECT)

调用接口获取登录凭证（code）进而换取用户登录态信息，包括用户的唯一标识（openid）及本次登录的会话密钥（session_key）。用户数据的加解密通信需要依赖会话密钥完成。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-78 所示。

表 B-78　wx.login(OBJECT) 中 OBJECT 参数说明

参数名	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-79 所示。

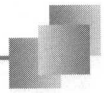


表 B-79　wx. login(OBJECT) success 返回参数说明

参 数 名	类　型	说　明
errMsg	String	调用结果
code	String	用户允许登录后，回调内容会带上 code（有效期五分钟），开发者需要将 code 发送到开发者服务器后台，使用 code 换取 session_key api，将 code 换成 openid 和 session_key

3）示例代码 B-59

```
//app.js
App({
  onLaunch:function(){
    wx.login({
      success:function(res){
        if (res.code){
          //发起网络请求
          wx.request({
            url:'https://test.com/onLogin',
            data:{
              code:res.code
            }
          })
        }else{
          console.log('获取用户登录态失败！'+res.errMsg)
        }
      }
    });
  }
})
```

2. code 换取 session_key

这是一个 HTTP 接口，开发者服务器使用登录凭证 code 获取 session_key 和 openid。其中 session_key 是对用户数据进行加密签名的密钥。为了自身应用安全，session_key 不应该在网络上传输。

1）接口地址

```
https://api.weixin.qq.com/sns/jscode2session?appid=APPID&secret=SECRET&js_code=JSCODE
&grant_type=authorization_code
```

2）请求参数

请求参数如表 B-80 所示。

表 B-80　请求参数

参　数	必　填	说　明
appid	是	小程序唯一标识
secret	是	小程序的 app secret
js_code	是	登录时获取的 code
grant_type	是	填写为 authorization_code

3）返回参数

返回参数如表 B-81 所示。

表 B-81　返回参数

参　数	说　明
openid	用户唯一标识
session_key	会话密钥
expires_in	会话有效期，以秒为单位，如 2592000 代表会话有效期为 30 天

4）返回说明

```
//正常返回的 JSON 数据包
{
        "openid":"OPENID",
        "session_key":"SESSIONKEY"
        "expires_in":2592000
}
//错误时返回 JSON 数据包(示例为 Code 无效)
{
    "errcode":40029,
    "errmsg":"invalid code"
}
```

3. 登录态维护

通过 wx. login( )获取到用户登录态之后，需要维护登录态。开发者要注意不应该直接把 session_key、openid 等字段作为用户的标识或者 session 的标识，而应该自己派发一个 session 登录态（见图 B-1）。对于开发者自己生成的 session，应该保证其安全性且不应该设置较长的过期时间。session 派发到小程序客户端之后，可将其存储在 storage，用于后续通信使用。

4. wx. checkSession(OBJECT)

1）返回参数说明

检查登陆态是否过期，返回参数说明如表 B-82 所示。

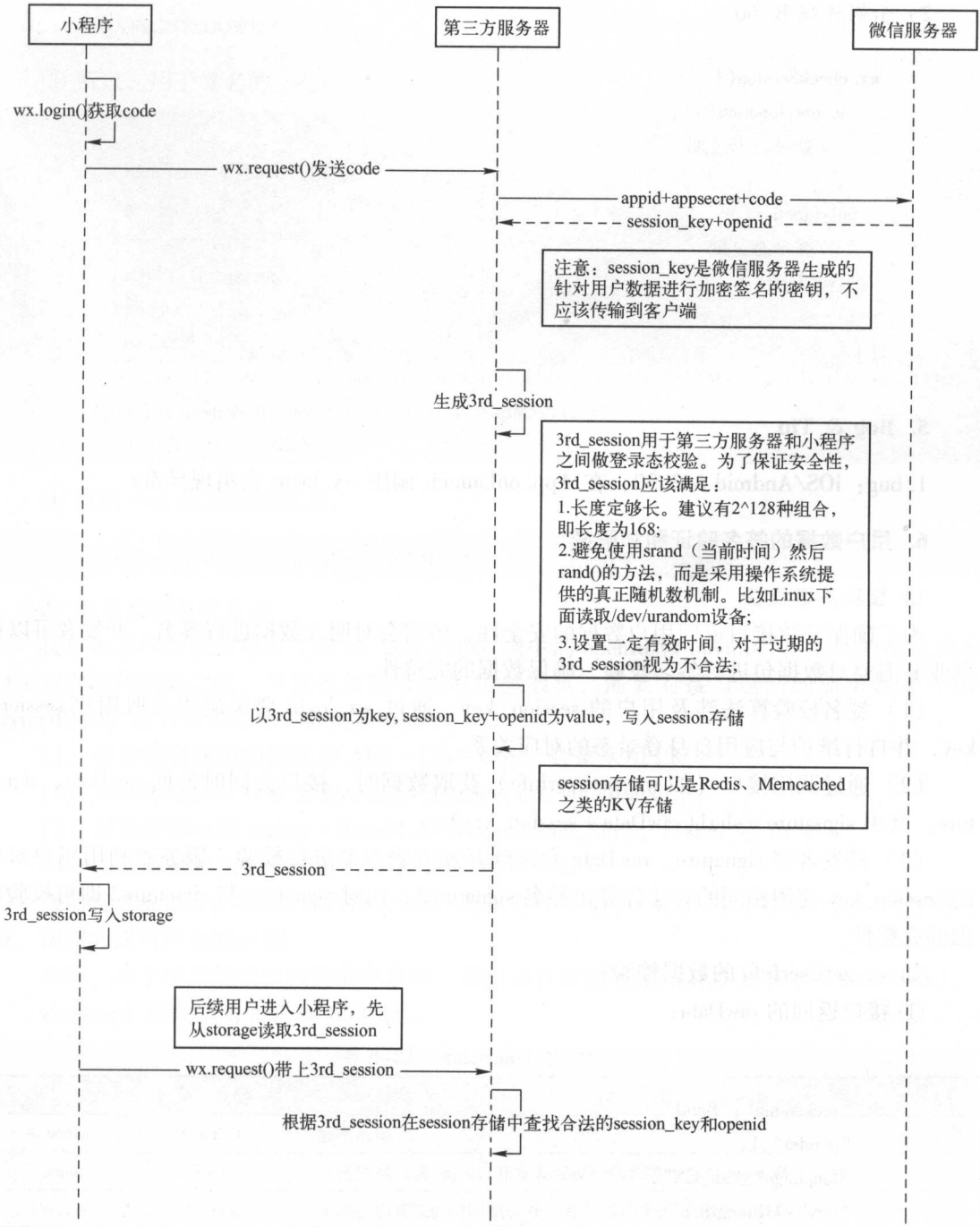


图 B-1　登录时序图

表 B-82　wx. checkSession(OBJECT)返回参数说明

参 数 名	类　型	必　填	说　明
success	Function	否	接口调用成功的回调函数，登录态未过期
fail	Function	否	接口调用失败的回调函数，登录态已过期
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）示例代码 B-60

```
wx.checkSession({
  success:function(){
    //登录态未过期
  },
  fail:function(){
    //登录态过期
    wx.login()
  }
})
```

5. Bug & Tip

1. bug：iOS/Android 6.3.30，在 App.onLaunch 调用 wx.login 会出现异常；

6. 用户数据的签名验证和加解密

1）数据签名校验

为了确保开放接口返回用户数据的安全性，微信会对明文数据进行签名。开发者可以根据业务需要对数据包进行签名校验，确保数据的完整性。

（1）签名校验算法涉及用户的 session_key，通过 wx.login 登录流程获取用户 session_key，并自行维护与应用自身登录态的对应关系。

（2）通过调用接口（如 wx.getUserInfo）获取数据时，接口会同时返回 rawData、signature，其中 signature = sha1(rawData + session_key)。

（3）开发者将 signature、rawData 发送到开发者服务器进行校验。服务器利用用户对应的 session_key 使用相同的算法计算出签名 signature2，比对 signature 与 signature2 即可校验数据的完整性。

如 wx.getUserInfo 的数据校验：

① 接口返回的 rawData：

```
{
  "nickName":"Band",
  "gender":1,
  "language":"zh_CN",
  "city":"Guangzhou",
  "province":"Guangdong",
  "country":"CN",
  "avatarUrl":"http://wx.qlogo.cn/mmopen/vi_32/1vZvI39NWFQ9XM4LtQpFrQJ1xlgZxx3w7bQxKARol6503Iuswjjn6nIGBiaycAjAtpujxyzYsrztuuICqIM5ibXQ/0"
}
```

② 用户的 session - key：



```
HyVFkGl5F5OQWJZZaNzBBg==
```

③ 所以，用于签名的字符串为：

```
{
  "nickName":"Band",
  "gender":1,
  "language":"zh_CN",
  "city":"Guangzhou",
  "province":"Guangdong",
  "country":"CN",
  "avatarUrl":"http://wx.qlogo.cn/mmopen/vi_32/1vZvI39NWFQ9XM4LtQpFrQJ1xlgZxx3w7bQxKARol6503Iuswjjn6nIGBiaycAjAtpujxyzYsrztuuICqIM5ibXQ/0"
}HyVFkGl5F5OQWJZZaNzBBg==
```

④ 使用 sha1 得到的结果为：

```
75e81ceda165f4ffa64f4068af58c64b8f54b88c
```

2）*加密数据解密算法*

接口如果涉及敏感数据（如 wx.getUserInfo 当中的 openId 和 unionId），接口的明文内容将不包含这些敏感数据。开发者如需要获取敏感数据，需要对接口返回的**加密数据**（encryptedData）进行对称解密。解密算法如下：

（1）对称解密使用的算法为 AES－128－CBC，数据采用 PKCS#7 填充。

（2）对称解密的目标密文为 Base64_Decode(encryptedData)。

（3）对称解密秘钥 aeskey = Base64_Decode(session_key)，aeskey 是 16 字节。

（4）对称解密算法初始向量 iv 会在数据接口中返回。

微信官方提供了多种编程语言的示例代码（单击下载）。每种语言类型的接口名字均一致。调用方式可以参照示例。

另外，为了应用能校验数据的有效性，我们会在敏感数据加上数据水印（watermark）。watermark 参数说明如表 B-83 所示。

表 B-83　watermark 参数说明

参　数	类　型	说　明
watermark	OBJECT	数据水印
appid	String	敏感数据归属 appid，开发者可校验此参数与自身 appid 是否一致
timestamp	DateInt	敏感数据获取的时间戳，开发者可以用于数据时效性校验

如接口 wx.getUserInfo 敏感数据当中的 watermark：

```
{
    "openId":"OPENID",
    "nickName":"NICKNAME",
    "gender":GENDER,
```

```
    "city":"CITY",
    "province":"PROVINCE",
    "country":"COUNTRY",
    "avatarUrl":"AVATARURL",
    "unionId":"UNIONID",
    "watermark":
    {
        "appid":"APPID",
        "timestamp":TIMESTAMP
    }
}
```

注：此前提供的加密数据（encryptData）及对应的加密算法将被弃用，请开发者不要再依赖旧逻辑。

B.8.2　用户信息

wx.getUserInfo(OBJECT)，获取用户信息，需要先调用 wx.login 接口。

1）OBJECT 参数说明

OBJECT 参数说明如表 B-84 所示。

表 B-84　wx.getUserInfo(OBJECT)中 OBJECT 参数说明

参数名	类型	必填	说明
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

2）success 返回参数说明

success 返回参数说明如表 B-85 所示。

表 B-85　wx.getUserInfo(OBJECT) success 返回参数说明

参数	类型	说明
userInfo	OBJECT	用户信息对象，不包含 openid 等敏感信息
rawData	String	不包括敏感信息的原始数据字符串，用于计算签名
signature	String	使用 sha1（rawData + sessionkey）得到字符串，用于校验用户信息
encryptedData	String	包括敏感数据在内的完整用户信息的加密数据，详细见加密数据解密算法
iv	String	加密算法的初始向量，详细见加密数据解密算法

3）示例代码 B-61

```
wx.getUserInfo({
  success:function(res){
    var userInfo = res.userInfo
    var nickName = userInfo.nickName
```

```
        var avatarUrl = userInfo.avatarUrl
        var gender = userInfo.gender //性别 0:未知、1:男、2:女
        var province = userInfo.province
        var city = userInfo.city
        var country = userInfo.country
    }
})
encryptedData 解密后为以下 json 结构,详见加密数据解密算法:
{
    "openId":"OPENID",
    "nickName":"NICKNAME",
    "gender":GENDER,
    "city":"CITY",
    "province":"PROVINCE",
    "country":"COUNTRY",
    "avatarUrl":"AVATARURL",
    "unionId":"UNIONID",
    "watermark":
    {
        "appid":"APPID",
    "timestamp":TIMESTAMP
    }
}
```

B.8.3　微信支付

wx.requestPayment(OBJECT)，发起微信支付。

1）Object 参数说明

Object 参数说明如表 B-86 所示。

表 B-86　wx.requestPayment(OBJECT)中 Object 参数说明

参　数	类　型	必　填	说　明
timeStamp	String	是	时间戳从 1970 年 1 月 1 日 00:00:00 至今的秒数，即当前的时间
nonceStr	String	是	随机字符串，长度为 32 个字符以下
package	String	是	统一下单接口返回的 prepay_id 参数值，提交格式如 prepay_id = *
signType	String	是	签名算法，暂支持 MD5
paySign	String	是	签名，具体签名方案参见微信公众号支付帮助文档
success	Function	否	接口调用成功的回调函数
fail	Function	否	接口调用失败的回调函数
complete	Function	否	接口调用结束的回调函数（调用成功、失败都会执行）

了解更多信息，请查看微信支付接口文档。

2）示例代码 B-62

```
wx. requestPayment( {
    'timeStamp ':'',
    'nonceStr ':'',
    'package ':'',
    'signType ':'MD5 ',
    'paySign ':'',
    'success ':function(res) {
    },
    'fail ':function(res) {
    }
})
```

B. 8. 4　模板信息

模板推送位置：服务通知。

模板下发条件：用户本人在微信体系内与页面有交互行为后触发。

模板跳转能力：单击查看详情仅能跳转下发模板的该账号的各个页面。

1. 获取模板 id

登录 https://mp. weixin. qq. com 获取模板，如果没有合适的模板，可以申请添加新模板，审核通过后可使用，详见模板审核说明。如图 B-2 所示。

（1）页面的 <form/ > 组件，属性 report - submit 为 true 时，可以声明为需发模板消息，此时单击按钮提交表单可以获取 formId，用于发送模板消息。或者当用户完成支付行为，可以获取 prepay_id 用于发送模板消息。

（2）调用接口下发模板消息（详见接口说明）。

2. 接口说明

1）获取 access_token

access_token 是全局唯一接口调用凭据，开发者调用各接口时都需使用 access_token，请妥善保存。access_token 的存储至少要保留 512 个字符空间。access_token 的有效期目前为 2 个小时，需定时刷新，重复获取将导致上次获取的 access_token 失效。

公众平台的 API 调用所需的 access_token 的使用及生成方式说明：

（1）为了保密 appsecrect，第三方需要一个 access_token 获取和刷新的中控服务器。而其他业务逻辑服务器所使用的 access_token 均来自于该中控服务器，不应该各自去刷新，否则会造成 access_token 覆盖而影响业务。

（2）目前 access_token 的有效期通过返回的 expires_in 来传达，目前是 7200 秒之内的



图 B-2　模板

值。中控服务器需要根据这个有效时间提前去刷新新 access_token。在刷新的过程中，中控服务器对外输出的依然是老 access_token，此时公众平台后台会保证在刷新短时间内，新老 access_token 都可用，这保证了第三方业务的平滑过渡。

（3）access_token 的有效时间可能会在未来有调整，所以中控服务器不仅需要内部定时主动刷新，还需要提供被动刷新 access_token 的接口，这样便于业务服务器在 API 调用获知 access_token 已超时的情况下，可以触发 access_token 的刷新流程。

开发者可以使用 AppID 和 AppSecret 调用本接口来获取 access_token。AppID 和 AppSecret 可登录微信公众平台官网 - 设置 - 开发设置中获得（需要已经绑定成为开发者，且账号没有异常状态）。AppSecret 生成后请自行保存，因为在公众平台每次生成查看都会导致 AppSecret被重置。注意调用所有微信接口时均需使用 https 协议。如果第三方不使用中控服务器，而是选择各个业务逻辑点各自去刷新 access_token，那么就可能会产生冲突，导致服务不稳定。

接口地址：

```
https://api.weixin.qq.com/cgi-bin/token?grant_type=client_credential&appid=APPID&secret=APPSECRET
```

HTTP 请求方式：GET

参数说明如表 B-87 所示。

表 B-87　GET 参数说明

参　数	必　填	说　明
grant_type	是	获取 access_token 填写 client_credential
appid	是	第三方用户唯一凭证
secret	是	第三方用户唯一凭证密钥，即 appsecret

正常情况下，微信会返回下述 JSON 数据包给开发者：

```
{"access_token":"ACCESS_TOKEN","expires_in":7200}
```

返回参数说明如表 B-88 所示。

表 B-88　返回参数说明

参　数	说　明
access_token	获取到的凭证
expires_in	凭证有效时间，单位：秒

错误时微信会返回错误码等信息，JSON 数据包示例如下（该示例为 AppID 无效错误）：

```
{"errcode":40013,"errmsg":"invalid appid"}
```

2）发送模板消息

接口地址：（ACCESS_TOKEN 需换成上文获取到的 access_token）

```
https://api.weixin.qq.com/cgi-bin/message/wxopen/template/send?access_token=ACCESS_TOKEN
```

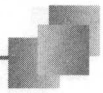
HTTP 请求方式：POST

POST 参数说明如表 B-89 所示。

表 B-89　POST 参数说明

参　数	必　填	说　明
touser	是	接收者（用户）的 openid
template_id	是	所需下发的模板消息的 id
page	否	单击模板查看详情跳转页面，不填则模板无跳转
form_id	是	表单提交场景下，为 submit 事件带上的 formId；支付场景下，为本次支付的 prepay_id
value	是	模板内容，不填则下发空模板
color	否	模板内容字体的颜色，不填默认黑色
emphasis_keyword	否	模板需要放大的关键词，不填则默认无放大

示例代码 B-63 如下所示。



```
{
  "touser":"OPENID",
  "template_id":"TEMPLATE_ID",
  "page":"index",
  "form_id":"FORMID",
  "data":{
      "keyword1":{
          "value":"339208499",
          "color":"#173177"
      },
      "keyword2":{
          "value":"2015 年 01 月 05 日 12:30",
          "color":"#173177"
      },
      "keyword3":{
          "value":"粤海喜来登酒店",
          "color":"#173177"
      },
      "keyword4":{
          "value":"广州市天河区天河路 208 号",
          "color":"#173177"
      }
  },
  "emphasis_keyword":"keyword1. DATA"
}
```

在调用模板消息接口后，会返回 JSON 数据包。

正常时的返回 JSON 数据包示例：

```
{
  "errcode":0,
  "errmsg":"ok",
}
```

错误时会返回错误码信息，说明如表 B-90 所示。

表 B-90　错误码信息说明

返　回　码	说　　明
40037	template_id 不正确
41028	form_id 不正确，或者过期
41029	form_id 已被使用
41030	page 不正确

3. 下发条件说明

1）支付

当用户在小程序内完成过支付行为，可允许开发者向用户在7天内推送有限条数的模板消息（1次支付可下发1条，多次支付下发条数独立，互相不影响）。

2）提交表单

当用户在小程序内发生过提交表单行为且该表单声明为要发模板消息的，开发者需要向用户提供服务时，可允许开发者向用户在7天内推送有限条数的模板消息（1次提交表单可下发1条，多次提交下发条数独立，相互不影响）。

4. Bug & Tip

Tip：微信6.5.2及以上版本支持模板功能。低于该版本将无法收到模板消息。

B.8.5　客服消息

1. 接收消息和事件

在页面中使用 <contact-button/> 可以显示进入客服会话按钮。

当用户在客服会话发送消息（或进行某些特定的用户操作引发的事件推送）时，微信服务器会将消息（或事件）的数据包（JSON或者XML格式）POST请求开发者填写的URL。开发者收到请求后可以使用发送客服消息接口进行异步回复。

微信服务器在将用户的消息发给小程序的开发者服务器地址（开发设置处配置）后，微信服务器在五秒内收不到响应会断掉连接，并且重新发起请求，总共重试三次，如果在调试中，发现用户无法收到响应的消息，可以检查是否消息处理超时。关于重试的消息排重，有msgid的消息推荐使用msgid排重。事件类型消息推荐使用FromUserName + CreateTime排重。

服务器收到请求如果不需要转发到客服工具，必须做出下述回复，这样微信服务器才不会对此做任何处理，并且不会发起重试。否则，将出现严重的错误提示。详见下面说明：

（1）直接回复success（推荐方式）；

（2）直接回复空串（指字节长度为0的空字符串，而不是结构体中content字段的内容为空）

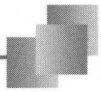
一旦遇到以下情况，微信都会在小程序会话中，向用户下发系统提示“该小程序客服暂时无法提供服务，请稍后再试”：

（1）开发者在5秒内未回复任何内容；

（2）开发者回复了异常数据。

如果开发者希望把消息或事件转发到客服工具，请参看转发消息到客服工具。客服工具接入请参考客服工具使用说明。

如果开发者希望增强安全性，可以在开发者中心处开启消息加密，这样，用户发给小程序的消息以及小程序被动回复用户消息都会继续加密，详见消息加解密说明。



各消息类型的推送 JSON、XML 数据包结构如下。

2. 文本消息

用户在客服会话中发送文本消息时将产生如下数据包。

1）XML 格式

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[text]]></MsgType>
    <Content><![CDATA[this is a test]]></Content>
    <MsgId>1234567890123456</MsgId>
</xml>
```

2）JSON 格式

```
{
    "ToUserName":"toUser",
    "FromUserName":"fromUser",
    "CreateTime":1482048670,
    "MsgType":"text",
    "Content":"this is a test",
    "MsgId":1234567890123456
}
```

3）参数说明

参数说明如表 B-91 所示。

表 B-91　参数说明

参　数	说　明
ToUserName	小程序的原始 ID
FromUserName	发送者的 openid
CreateTime	消息创建时间（整型）
MsgType	text
Content	文本消息内容
MsgId	消息 id，64 位整型

3. 图片消息

用户在客服会话中发送图片消息时将产生如下数据包。

1）XML 格式

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[image]]></MsgType>
    <PicUrl><![CDATA[this is a url]]></PicUrl>
    <MediaId><![CDATA[media_id]]></MediaId>
    <MsgId>1234567890123456</MsgId>
</xml>
```

2）JSON 格式

```
{
    "ToUserName":"toUser",
    "FromUserName":"fromUser",
    "CreateTime":1482048670,
    "MsgType":"image",
    "PicUrl":"this is a url",
    "MediaId":"media_id",
    "MsgId":1234567890123456
}
```

3）参数说明

参数说明如表 B-92 所示。

表 B-92　参数说明

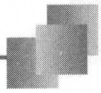
参　数	说　明
ToUserName	小程序的原始 ID
FromUserName	发送者的 openid
CreateTime	消息创建时间（整型）
MsgType	image
Content	图片链接（由系统生成）
MediaId	图片消息媒体 id，可以调用获取临时素材接口拉取数据
MsgId	消息 id，64 位整型

4. 进入会话事件

用户在小程序“客服会话按钮”进入客服会话时将产生如下数据包。

1）XML 格式

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
```



```
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[event]]></MsgType>
    <Event><![CDATA[user_enter_tempsession]]></Event>
    <SessionFrom><![CDATA[sessionFrom]]></SessionFrom>
</xml>
```

2）JSON 格式

```
{
    "ToUserName":"toUser",
    "FromUserName":"fromUser",
    "CreateTime":1482048670,
    "MsgType":"event",
    "Event":"user_enter_tempsession",
    "SessionFrom":"sessionFrom"
}
```

3）参数说明

参数说明如表 B-93 所示。

表 B-93　参数说明

参　数	说　明
ToUserName	小程序的原始 ID
FromUserName	发送者的 openid
CreateTime	事件创建时间（整型）
MsgType	event
Event	事件类型，user_enter_tempsession
SessionFrom	开发者在客服会话按钮设置的 sessionFrom 参数

5. 转发消息到客服工具

如果开发者希望消息被转发到客服工具处理，请按照下述结构返回响应包。响应包的数据格式（XML 或 JSON）需要与设置的接收消息数据格式一致。

1）XML 格式

```
<xml>
    <ToUserName><![CDATA[toUser]]></ToUserName>
    <FromUserName><![CDATA[fromUser]]></FromUserName>
    <CreateTime>1482048670</CreateTime>
    <MsgType><![CDATA[transfer_customer_service]]></MsgType>
</xml>
```

2）JSON 格式

```
{
    "ToUserName":"toUser",
    "FromUserName":"fromUser",
    "CreateTime":1482048670,
    "MsgType":"transfer_customer_service",
}
```

3）参数说明

参数说明如表 B-94 所示。

表 B-94　参数说明

参　数	说　明
ToUserName	发送者的 openid
FromUserName	小程序原始 ID
CreateTime	事件创建时间（整型）
MsgType	transfer_customer_service

6. 发送客服消息

当用户和小程序客服产生特定动作的交互时（具体动作列表请见下方说明），微信将会把消息数据推送给开发者，开发者可以在一段时间内（目前修改为 48 小时）调用客服接口，通过 POST 一个 JSON 数据包来发送消息给普通用户。此接口主要用于客服等有人工消息处理环节的功能，方便开发者为用户提供更加优质的服务。

目前允许的动作如表 B-95 所示，不同动作触发后，允许的客服接口下发消息条数和下发时限不同。下发条数达到上限后，会收到错误返回码，具体请见返回码说明页。

表 B-95　允许的动作

用 户 动 作	允许下发条数限制	下 发 时 限
用户通过客服消息按钮进入会话	1 条	1 分钟
用户发送信息	3 条	48 小时

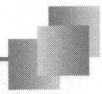
客服接口 - 发消息，接口调用请求说明：

```
http 请求方式:POST
https://api.weixin.qq.com/cgi-bin/message/custom/send?access_token=ACCESS_TOKEN
```

各消息类型所需的 JSON 数据包如下。

1）发送文本消息

```
{
    "touser":"OPENID",
    "msgtype":"text",
    "text":
    {
        "content":"Hello World"
    }
}
```



2）发送图片消息

```
{
    "touser":"OPENID",
    "msgtype":"image",
    "image":
    {
        "media_id":"MEDIA_ID"
    }
}
```

3）参数说明

参数说明如表 B-96 所示。

表 B-96　参数说明

参　数	是否必须	说　明
access_token	是	调用接口凭证
touser	是	普通用户（OPENID）
msgtype	是	消息类型，文本为 text，图片为 image
content	是	文本消息内容
media_id	是	发送的图片的媒体 ID，通过新增素材接口上传图片文件获得

4）返回码说明

返回码说明如表 B-97 所示。

表 B-97　返回码说明

参　数	说　明
-1	系统繁忙，此时请开发者稍候再试
0	请求成功
40001	获取 access_token 时 AppSecret 错误，或者 access_token 无效。请开发者认真比对 AppSecret 的正确性，或查看是否正在为恰当的小程序调用接口
40002	不合法的凭证类型
40003	不合法的 OpenID，请开发者确认 OpenID 否是其他小程序的 OpenID
45015	回复时间超过限制
45047	客服接口下行条数超过上限
48001	api 功能未授权，请确认小程序已获得该接口

7. 临时素材接口

1）获取临时素材

小程序可以使用本接口获取客服消息内的临时素材（即下载临时的多媒体文件）。目前小程序仅支持下载图片文件。

（1）接口调用请求说明如下。

① HTTP 请求方式：GET、HTTPS 调用。

```
https://api.weixin.qq.com/cgi-bin/media/get? access_token=ACCESS_TOKEN&media_id=MEDIA_ID
```

② 请求示例（示例为通过 curl 命令获取多媒体文件）：

```
curl -I -G "https://api.weixin.qq.com/cgi-bin/media/get? access_token=ACCESS_TOKEN&media_id=MEDIA_ID"
```

（2）参数说明如表 B-98 所示。

表 B-98　参数说明

参　数	是否必须	说　明
access_token	是	调用接口凭证
media_id	是	媒体文件 ID

（3）返回说明如下所示。

正确情况下的返回 HTTP 头如下：

```
HTTP/1.1 200 OK
Connection:close
Content-Type:image/jpeg
Content-disposition:attachment;filename="MEDIA_ID.jpg"
Date:Sun,06 Jan 2013 10:20:18 GMT
Cache-Control:no-cache,must-revalidate
Content-Length:339721
curl -G "https://api.weixin.qq.com/cgi-bin/media/get? access_token=ACCESS_TOKEN&media_id=MEDIA_ID"
```

如果返回的是视频消息素材，则内容如下：

```
{
  "video_url":DOWN_URL
}
```

错误情况下的返回 JSON 数据包示例如下（示例为无效媒体 ID 错误）：

```
{
  "errcode":40007,
  "errmsg":"invalid media_id"
}
```

2）新增临时素材

小程序可以使用本接口把媒体文件（目前仅支持图片）上传到微信服务器，用户发送

客服消息或被动回复用户消息。

（1）接口调用请求说明如下。

① HTTP 请求方式：POST/FORM、HTTPS 调用。

```
https://api.weixin.qq.com/cgi-bin/media/upload?access_token=ACCESS_TOKEN&type=TYPE
```

② 调用示例（使用 curl 命令，用 FORM 表单方式上传一个多媒体文件）。

```
curl -F media=@test.jpg "https://api.weixin.qq.com/cgi-bin/media/upload?access_token=ACCESS_TOKEN&type=TYPE"
```

（2）参数说明如表 B-99 所示。

表 B-99　参数说明

参　数	是否必须	说　明
access_token	是	调用接口凭证
type	是	image
media	是	form-data 中媒体文件标识，有 filename、filelength、content-type 等信息

（3）返回说明如下。

正确情况下的返回 JSON 数据包结果如下：

```
{
    "type":"TYPE",
    "media_id":"MEDIA_ID",
    "created_at":123456789
}
```

返回参数说明如表 B-100 所示。

表 B-100　返回参数说明

参　数	描　述
type	image
media_id	媒体文件上传后，获取标识
created_at	媒体文件上传时间戳

错误情况下的返回 JSON 数据包示例如下（示例为无效媒体类型错误）：

```
{
    "errcode":40004,
    "errmsg":"invalid media type"
}
```

B.8.6 分享

在 Page 中定义 onShareAppMessage 函数，设置该页面的分享信息。

- 只有定义了此事件处理函数，右上角菜单才会显示“分享”按钮
- 用户点击分享按钮的时候会调用
- 此事件需要 return 一个 Object，用于自定义分享内容

1. 自定义分享字段

自定义分享字段如表 B-101 所示。

表 B-101 自定义分享字段

字 段	说 明	默 认 值
title	分享标题	当前小程序名称
desc	分享描述	当前小程序名称
path	分享路径	当前页面 path，必须是以/开头的完整路径

2. 示例代码 B-64

```
Page({
  onShareAppMessage:function (){
    return{
      title:'自定义分享标题',
      desc:'自定义分享描述',
      path:'/page/user? id=123'
    }
  }
})
```

附录 C　JavaScript 的关键字和操作符

C. 1　JavaScript 操作符

1. 算术操作符（表 C-1）

表 C-1　算术操作符

*	两个数相乘
/	两个数相除
%（模）	两个数进行整数除法之后的余数

2. 字符串操作符（表 C-2）

表 C-2　字符串操作符

+	（字符串相加）连接两个字符串
+=	连接两个字符串，并且把得到的字符串赋予第一个操作对象

3. 逻辑操作符（表 C-3）

表 C-3　逻辑操作符

&&	（逻辑与）如果两个操作数都是 true，就返回 true；否则返回 false
||	（逻辑或）只要有一个操作数是 true，就返回 true；如果两个操作数都是 false，就返回 false
!	（逻辑非）如果操作数是 true，就返回 false；如果操作数是 false，就返回 true

4. 位操作符（表 C-4）

表 C-4　位操作符

&	（位与）如果操作数的对应位都是 1，返回值的相应位就是 1
^	（位异或）如果操作数的对应位中只有一个是 1，返回值的相应位就是 1
|	（位或）如果操作数的对应位中有一个是 1，返回值的相应位就是 1
~	（位非）把 1 的位置变成 0，把 0 的位置变成 1，也就是把每一位都取反
<<	（左移位）把第一个操作数向左移动第二个操作数指定的位数
<<	（符号填充的右移位）把第一个操作数向右移动第二个操作数指定的位数
>>>	（0 填充的右移位）把第一个操作数向右移动第二个操作数指定的位数，同时从左方移入 0

5. 赋值操作符（表 C-5）

表 C-5　赋值操作符

=	如果第一个操作数是变量，把第二个操作数的值赋予它
+=	如果第一个操作数是变量，就把两个操作数相加，把结果赋予它
-=	如果第一个操作数是变量，就把两个操作数相减，把结果赋予它
*=	如果第一个操作数是变量，就把两个操作数相乘，把结果赋予它
/=	如果第一个操作数是变量，就把两个操作数相除，把结果赋予它
%=	如果第一个操作数是变量，就把两个操作数进行模运算，把结果赋予它
&=	如果第一个操作数是变量，就把两个操作数进行位与运算，把结果赋予它
^=	如果第一个操作数是变量，就把两个操作数进行位异运算，把结果赋予它
|=	如果第一个操作数是变量，就把两个操作数进行位或运算，把结果赋予它
<<=	如果第一个操作数是变量，就把两个操作数进行左移位操作，把结果赋予它
>>=	如果第一个操作数是变量，就把两个操作数进行右移位操作，把结果赋予它
>>>=	如果第一个操作数是变量，就把两个操作数进行填0的右移位操作，把结果赋予它

6. 比较操作符（表 C-6）

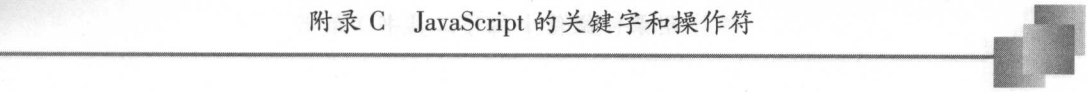
表 C-6　比较操作符

==	（相等操作符）如果两个操作数彼此相等，返回 true
!=	（不相等操作符）如果两个操作数不相等，返回 true
===	（严格相等）如果两个操作数类型相同、值相等，返回 true
!==	（严格不相等）如果两个操作数或者类型不相同，或者值不相等，返回 true
>	（大于）如果第一个操作数的值大于第二个操作数，返回 true
>=	（大于等于）如果第一个操作数的值大于或等于第二个操作数，返回 true
<	（小于）如果第一个操作数的值小于第二个操作数，返回 true
<=	（小于等于）如果第一个操作数的值小于或等于第二个操作数，返回 true

7. 特殊操作符（表 C-7）

表 C-7　特殊操作符

?:	（条件操作符）执行像“if...else”的操作
,	（逗号操作符）计算两个表达式的值，返回第二个表达式的值
delete	（删除）删除一个对象，并且把它从内存移除；或是删除对象的属性，或是删除数组里的一个元素
function	创建匿名函数
in	如果指定对象包含特定属性，就返回 true
操作符	描述
instanceof	如果指定对象是特定类型的实例，就返回 true
new	创建指定类型的一个新对象
typeof	返回操作数的类型名称
void	允许计算表达式的值但不返回值



C.2　字符串方法（表 C-8）

表 C-8　字符串方法

方　法	描　述
substring	返回字符串的一部分
toUpperCase	把字符串里全部字符转化为大写
toLowerCase	把字符串里全部字符转化为小写
indexof	在一个字符串里寻找另一个字符串
lastIndexof	从一个字符串的末尾开始寻找另一个字符串
replace	在字符串里寻找特定子串，用新的子串替代找到的子串
slipt	把字符串分解为一个数组，返回一个新数组
link	利用字符串的文本创建一个 HTML 链接
anchor	在当前页面里创建一个 HTML 锚点

C.3　Math 对象

1. 常数（表 C-9）

表 C-9　常数

E	自然对数的底（大约是 2.718）
LN2	2 的自然对数（大约是 0.693）
LN10	10 的自然对数（大约是 2.302）
LOG2E	以 2 为底 e 的对数（大约是 1.442）
LOG10E	以 10 为底 e 的对数（大约是 0.434）
PI	圆周率（大约是 3.141 59）
SQRT1_2	0.5 的平方根（大约是 0.707）
SQRT2	2 的平方根（大约是 1.414 2）

2. 代数方法（表 C-10）

表 C-10　代数方法

acos	反余弦
asin	反正弦
atan	反正切
cos	余弦
sin	正弦
tan	正切

3. 统计与对数（表 C-11）

表 C-11　统计与对数

exp	返回 e（自然对数）的幂
log	返回数值的自然对数
max	接收两个数值，返回较大的数值
min	接收两个数值，返回较小的数值

4. 基本运算和取整（表 C-12）

表 C-12　基本运算和取整

abs	数值的绝对值
ceil	把数值向上取整到最近的整数
floor	把数值向下取整到最近的整数
pow	一个数值与另一个数值的幂
round	把数值取整到最近的整数
sqrt	数值的平方根
random	0 到 1 之间的随机数

C.4　Date 对象（表 C-13）

表 C-13　Date 对象

getDate( )	返回月份里的日期（1 ~ 31）
getDay( )	返回星期里的日期（0 ~ 6）
getFullYear( )	返回年份（四位数字）
getHours( )	返回小时数（0 ~ 23）
getMilliseconds( )	返回毫秒数（0 ~ 999）
getMinutes( )	返回分钟数（0 ~ 59）
getMonth( )	返回月份数（0 ~ 11）
getSeconds( )	返回秒数（0 ~ 59）
getTime( )	获得自 1970 年 1 月 1 日以来的毫秒数
getTimezoneOffset( )	返回与 GMT 的时差及本地时间，单位是分
getUTCDate( )	根据全球统一时间返回月份里的日期（1 ~ 31）
getUTCDay( )	根据全球统一时间返回星期里的日期（0 ~ 6）
getUTCFullYear( )	根据全球统一时间返回年份（四位数字）
getUTCHours( )	根据全球统一时间返回小时数（0 ~ 23）
getUTCMilliseconds( )	根据全球统一时间返回毫秒数（0 ~ 999）

续表

getUTCMinutes( )	根据全球统一时间返回分钟数（0 ~ 59）
getUTCMonth( )	根据全球统一时间返回月份数（0 ~ 11）
getUTCSeconds( )	根据全球统一时间返回秒数（0 ~ 59）
parse( )	解析日期字符串，返回自 1970 年 1 月 1 日午夜至今的毫秒数
setDate( )	设置月份里的日期数（1 ~ 31）
setFullYear( )	设置年份（四位数字）
setHours( )	设置小时数（0 ~ 23）
setMilliseconds( )	设置毫秒数（0 ~ 999）
setMinutes( )	设置分钟数（0 ~ 59）
setMonth( )	设置月份数（0 ~ 11）
setSeconds( )	设置秒数（0 ~ 59）
setTime( )	对自 1970 年 1 月 1 日午夜以来的毫秒数进行加减，以结果设置日期和时间
setUTCDate( )	根据全球统一时间设置月份里的日期（1 ~ 31）
setUTCFullYear( )	根据全球统一时间设置年份（四位数字）
setUTCHours( )	根据全球统一时间设置小时数（0 ~ 23）
setUTCMilliseconds( )	根据全球统一时间设置毫秒数（0 ~ 999）
setUTCMinutes( )	根据全球统一时间设置分钟数（0 ~ 59）
setUTCMonth( )	根据全球统一时间设置月份数（0 ~ 11）
setUTCSeconds( )	根据全球统一时间设置秒数（0 ~ 59）
toDateString( )	把 Date 对象的日期部分转化为直观的字符串
toLocaleDateString( )	以遵循本地规范的字符串返回 Date 对象的日期部分
toLocaleTimeString( )	以遵循本地规范的字符串返回 Date 对象的时间部分
toLocaleString( )	以遵循本地规范的字符串返回 Date 对象
toString( )	把 Date 对象转化为字符串
toTimeString( )	把 Date 对象的时间部分转化为字符串
toUTCString( )	根据全球统一时间把 Date 对象转化为字符串
UTC( )	根据全球统一时间，返回自 1970 年 1 月 1 日午夜以来的毫秒数
valueOf( )	返回 Date 对象的原始值

■ 作者简介

老王，十年软件开发和互联网产品运营推广经验，“ 打字学英语 ”App 创始人，思行合一科技有限公司 CEO，微信小程序俱乐部专栏作家，天使投资人。

新浪微博：@ 老王的故事 001

微信：2264538870

- 一本从零开始学微信小程序开发的书
- 微信小程序开发从入门到精通就它了
- 详细解读JavaScript、WXML、CSS语法
- 前端设计开发人员必备案头图书

电子信息出版分社微博
http://weibo.com/etpublish
http://www.weibo.com/wjd80

策划编辑：张　迪
责任编辑：张　迪
封面设计：徐海燕

官方微信平台

ISBN 978-7-121-30850-5

定价：49.00元